SUPER BOOKS 2
PC-8801mkⅡSR
やさしいマシン語
前田光男 著
技術評論社

SUPER BOOKS 2

# PC-8801mkⅡSR

# やさしいマシン語

前田光男 著

技術評論社

# はじめに

1985年1月に発表されたPC-8801mkIISRは，PC88シリーズの3代目にあたります．SRは高解像度高速多機能グラフィックスの完備やFM音源ユニットの搭載などの性能を持つ，8ビットとしては完成度の高いパソコンです．これらの優れた性能を100％引き出すには，BASICだけでは不可能です（せいぜい60％といったところでしょうか）．それは，BASICではハードウェアに関する細かいプログラミングができないからです．もし，BASICで細かいプログラミングをしたいのであれば，BASICコマンドの中にあるマシン語を操作するコマンド（USR，POKE，PEEKなど）を使用しなければなりません．当然，これらのBASICコマンドを使うためにはマシン語の知識が必要となります．

またBASICはインタプリタなので実行速度が遅いという欠点を持っています．実行速度の速いマシン語を使うと，この欠点もなくなります．

本書はマシン語をやさしく，かつ，より深く理解できるように執筆したもので，構成もこれに沿っています．前半はマシン語の基礎から入ってだんだんと理解を深めるようにしてあります．そして，後半の「マシン語アラカルト」で前半で説明できなかったことや，筆者の勤務先の科学部の生徒達が日頃疑問に思っていることについて説明しています．

マシン語を早く覚えるコツは，本書のプログラムを理解したら一部を変えたり，同じ動きをするプログラムを別に作ったりしてゆくことです．読者の皆さんが早くマシン語を理解し，SRの性能を100％引き出せることを期待します．

最後に，執筆のお勧めを頂き，遅筆の筆者を絶えず励まして下さった技術評論社の編集スタッフ諸氏に感謝申し上げる次第です．

1985年　7月　　　　前田　光男

# CONTENTS

## 第1章 初級者から中級者までのマシン語
～誰もが知っておきたい基礎知識

## 第3章 キーボードを操作する

~キーボードからデータを読み取る

## 第4章 グラフィックスの世界
～グラフィックス・パターンの作り方と移動方法

## 第5章 スムーズ・スクロール
~横方向をドット単位で動かす

# マシン語アラカルト

# 付録

第1章

# 初級者から中級者までのマシン語

誰もが知っておきたい基礎知識

「マシン語って難しい」

「いろいろ本を読んだけどよくわからなかった」

こんな話をよく聞きます．マシン語は他の言語(BASICやFORTRAN)などとは較べものにならないほどレベル差の激しい言語です．ですから，レベルに合った知識が必要で，初めての人はまずマシン語の大枠をつかむことが大切です．

そこでこの章では，マシン語を勉強していく上で必要な最小限の知識をコンパクトにまとめてみました．初めての人はとにかくこの章をじっくり読んでください．そうでない人も，ざっと自分の知識を再確認して，足りないところを補っておきましょう．

*Section 1*

# 2進数と16進数

Base

## 2進数は0と1の世界

コンピュータの世界は，０と１の２通りから成り立っています．したがって，コンピュータに命令を与えたりするときは，０と１だけの組み合わせ，すなわち２進数で行うことになります．**表1-1**に２進数と10進数の対応表を示します．

２進数では０と１しかありませんから，１桁で表せる数は０と１だけです．これが２桁になると00, 01, 10, 11の組み合わせがありますから，０，１，２，３と４つの数を表現できるようになります．同じように３桁では８つの数が表現できます．桁数が増えれば増えるほど，表現できる数は多くなります．一般に桁数と表現できる数との関係は次の式で表されます．

**$2^n$＝表すことのできる数(n は桁数を表す)**

たとえば２桁のときは，

$2^n : 2^2 = 2\times2 = 4$

で４個の数を表現できます．４桁のときは，

$2^n : 2^4 = 2\times2\times2\times2 = 16$

で16個の数を表現できることになります．

**表1-1　2進数と10進数**

| 2進数 | 10進数 |
|---|---|
| 0000 | 0 |
| 0001 | 1 |
| 0010 | 2 |
| 0011 | 3 |
| 0100 | 4 |
| 0101 | 5 |
| 0110 | 6 |
| 0111 | 7 |
| 1000 | 8 |
| 1001 | 9 |
| 1010 | 10 |
| 1011 | 11 |
| 1100 | 12 |
| 1101 | 13 |
| 1110 | 14 |
| 1111 | 15 |

### ■ビットとは２進数の桁のこと

２進数の桁のことをビットと言います．PC-8801mkIISR は８ビットマシンと呼ばれていますから，一度に扱える２進数の桁は８桁です．８ビットで表現できる数は，

$2^8 = 256$

なので PC-8801mkIISR では，一度に256個の数を表現できるわけです．８ビッ

トマシンでは，この256までの数値を使用してコンピュータに命令しています．なお，これからは2進数の桁のことをビットと呼んでいくことにしましょう．ビットの数と表現できる数との関係を**表1-2**に示します．

**表1-2**
**ビットの数と表現できる数**
※○印はよく使用されるビット

| ビット | 数 | ビット | 数 |
|---|---|---|---|
| 1ビット → | 2 | 9ビット → | 512 |
| 2ビット → | 4 | ○10ビット → | 1024 |
| 3ビット → | 8 | 11ビット → | 2048 |
| 4ビット → | 16 | 12ビット → | 4096 |
| 5ビット → | 32 | 13ビット → | 8192 |
| 6ビット → | 64 | 14ビット → | 16384 |
| 7ビット → | 128 | 15ビット → | 32768 |
| ○8ビット → | 256 | ○16ビット → | 65536 |

10進数では，一，十，百，千と桁ごとに位がついていました．同じように，2進数も桁ごとに位を持っています．この2進数の桁の位のことを重みと言う場合があります．8ビットを例にとって，桁の位を見てみると**図1-1**のようになります．これを見ると，2進数を10進数に直すには，1になっているビットの位の重みを足していけば良いことがわかります．ビット数が多くなっても同じことです．

**図1-1　2進数を10進数に直す**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 2進数8ビット | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | |
| 2進数の桁の位 | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ | |
| *( )の中は10進数の重み | (128) | (64) | (32) | (16) | (8) | (4) | (2) | (1) | |
| 10進数に直すと | $1\times2^7$ | $0\times2^6$ | $1\times2^5$ | $0\times2^4$ | $0\times2^3$ | $0\times2^2$ | $1\times2^1$ | $1\times2^0$ | |
| | 128+ | 0 + | 32+ | 0 + | 0 + | 0 + | 2 + | 1 | = 163 |

## ビットとバイト

2進数は0と1を使用しているので，よく10進数の数値と間違われます．そこで10進数と区別するために，2進数の数値の最後に“B”(Binary)をつけます．たとえば，

　　1001B　　(読み方　イチ，ゼロ，ゼロ，イチ)

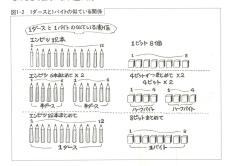
これは2進数ですから，当然，読み方も10進数とは違います．2進数には0と

1しかないので，例のように「イチ，ゼロ，ゼロ，イチ」と読むことになります．しかし，ビット数が大きくなって8ビットや16ビットになると，何のことを言っているのかわからなくなります．これでは大変です．

**■8ビット＝1バイト**

そこで，何ビットかずつ区切って，これをひとつの単位として扱うことが考え出されました．現在は8ビットをひとつの単位（1バイト）として扱っています．そして1バイトの半分，すなわち4ビットのことをハーフバイトと呼んでいます．昔は1バイトが5ビットとか6ビットとかまちまちでしたが，コンピュータが発達してからは，文字の記憶に便利な8ビットに定着しました．

8ビットを1バイトと呼んでいるのは，エンピツ12本を1ダースと呼んでいるのとよく似ています(図1-2)．

図1-2　1ダースと1バイトの似ている関係

## 16進数を使って2進数を読みやすく

8ビットを1バイトと呼ぶのはよいのですが，呼び方が「イチ，ゼロ，イチ……」では今までと何ら変りありません．そこで，わかりやすくするために1バイトを4ビット，つまりハーフバイトずつ区切って読む方法があります．4

ビットで表せる数は全部で16です．そこで，16種類の文字からなる16進数を使って１バイトを読むとわかりやすいのです．

16進数は，0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，Fの16種類の文字から成り立っています．A，B，C，D，E，Fはアルファベットの文字ですが，16進数では数字の意味ですので注意してください．昔は，W，Z，……などという文字も使用されていましたが，現在はすべてA～Fの文字に統一されています．

## ■“H”をつけると16進数

また，16進数でも０から９までは10進数の文字を使用しているため，10進数か16進数か区別できません．そこで16進数を10進数とはっきり区別して表すために，２進数のときと同じ方法で16進数の数値の最後に“H”（Hexadecimal）をつけています．たとえば，

8230H

となります．

さらに，16進数では０～Fまでの文字を使用しているために，A～Fが先頭にくる場合もあります．もし，A～Fが先頭にきたときに16進数の数値であることを強調したい場合は，A～Fの文字の前に“0”をつけます．

0F3C0H

この表記法は特にアセンブリ言語を使用したときに，ラベルと数値を区別するために使用されます．アセンブラにかけたプログラムリストを見るときや，

表1-3
2進数，10進数，16進数

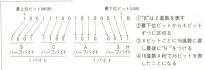
| 2進数 | 10進数 | 16進数 |
|---|---|---|
| 0000B | 0 | 0H |
| 0001B | 1 | 1H |
| 0010B | 2 | 2H |
| 0011B | 3 | 3H |
| 0100B | 4 | 4H |
| 0101B | 5 | 5H |
| 0110B | 6 | 6H |
| 0111B | 7 | 7H |
| 1000B | 8 | 8H |
| 1001B | 9 | 9H |
| 1010B | 10 | AH |
| 1011B | 11 | BH |
| 1100B | 12 | CH |
| 1101B | 13 | DH |
| 1110B | 14 | EH |
| 1111B | 15 | FH |

雑誌などのリストを見るときにでてきますね．

2進数，10進数，16進数をまとめると，P.18の**表1-3**のようになります．

## ■ 2進数から16進数へ

16進数は4ビットデータをちょうど1桁で表すことができるので，8ビットや16ビットをそれぞれ2桁，4桁で表現できます．2進数を16進数に直す場合，**図1-3**のようになります．

**図1-3　2進数を16進数に直す**

2進数で表すよりも16進数で表した方が，数値の大小関係がわかりやすいですね．この方が覚えやすいし，間違いも少なくなります．さて，こうなると2進数など必要ないようですが，そうはいきません．たとえば，各ビットごとに意味のある場合などは16進数を見てもピンときませんが，2進数ならばすぐにわかります．PC-8801mkIISRでは，グラフィックスにビット・マップ法（グラフィックスの章で説明します）を採用しているために，2進数が絶対に必要となってくるのです．2進数と16進数の復習を簡単にやってみましたが，いかがでしたか？　まあ，忘れていたひとはもう一度読み直してください．

*Section 2*

# Z-80A CPU

## Z-80Aは8ビットCPUの主流

8ビットCPUを使用したパソコンは数多く発売されていますが，その多くで"Z-80A"というCPUが使われています．Z-80Aは，8ビットCPUの主流といえるかもしれませんね．CPUとは，中央演算処理装置(Central Processing Unit)のことで，コンピュータの頭脳にあたるもので，すべての計算は必ずこの部分で行われます．

現在，マイコンのCPUは，

**8080系 CPU（インテル社）**

**6800系 CPU（モトローラ社）**

の2つに大きく分かれています．なお，8080というCPUはZ-80A(ザイログ社)に取って代られており，6800も6809(モトローラ社)に取って代られています．PC-8801mkIISRには，このZ-80Aが使用されています．

Z-80A CPUは，40ピンのプラスチック(またはセラミック)の筐体の中に入っていて，その本体は5～6㎜四方の小さいものです．したがって，CPUを直接見ることはできません．

CPUの内部は見ることができませんが，プログラムに関することはマニュアルで知ることができます．プログラムに関係するレジスタは約20個しかありませんが，この約20個のために100種類を越す命令が用意されています．

## Z-80Aの内部構成

マシン語を学ぶ際には，CPUの内部構成を覚える必要があります．CPUの内部構成は**図1-4**のようになっています．

それでは，CPUについて簡単に説明していきましょう．まず，レジスタとは，CPUの内部にある特別な働きをするメモリです．CPUを構成しているレジス

タは，それぞれ特徴を持っていて各々に名前がつけられています．各々のレジスタとその働きをまとめてみましょう．

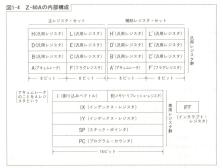
## ■主レジスタ・セット

自由に使用できるレジスタで，計算を行うなど用途がたくさんあります．

| | |
|---|---|
| **Hレジスタ** | **Lレジスタ** |
| **Dレジスタ** | **Eレジスタ** |
| **Bレジスタ** | **Cレジスタ** |
| **Aレジスタ** | **F（フラグ）レジスタ** |

この中で最も大切なのは，Aレジスタ（アキュムレータ）とF（フラグ）レジスタです．Aレジスタは特別な機能を持っていて，すべての計算の中心になる部分なので特別にアキュムレータと呼ぶこともあります．次にF(フラグ)レジスタですが，これは汎用レジスタ（H，L，D，E，B，C，Aレジスタのこと）の状態を表すものです．プログラム上の分岐はすべてこのレジスタのお世話になるので，非常に大切なレジスタです．また，AレジスタとF(フラグ)レジスタをあわせてPSW（Program Status Word)ということもあります．

汎用レジスタは，それぞれ8ビットのレジスタですが，次のように組み合わ

**図1-4 Z-80Aの内部構成**

せると16ビットレジスタとしても使用できます．

| (上位バイト) | | (下位バイト) | | |
|---|---|---|---|---|
| Hレジスタ | ＋ | Lレジスタ | ＝ | HLペアレジスタ |
| Dレジスタ | ＋ | Eレジスタ | ＝ | DEペアレジスタ |
| Bレジスタ | ＋ | Cレジスタ | ＝ | BCペアレジスタ |
| (8ビット) | ＋ | (8ビット) | ＝ | (16ビット) |

なお，16ビットレジスタとして使用する際には，上記以外の組み合わせ（たとえばHレジスタとBレジスタ）では使用できません．16ビットレジスタとしての使用は，アドレス（番地）などに数多く使われます．

## ■補助レジスタ・セット（副レジスタ・セット）

この補助レジスタセットは，すべての面で主レジスタ・セットと全く同じです．補助レジスタ・セットは，主レジスタ・セットのデータの一時的な保存に使用します．しかし，その際CPUは主レジスタ・セットを使用しているのか，補助レジスタ・セットを使用しているのか情報を出しません．そこで使用方法をはっきりさせておく必要があります．

**▼SP（スタック・ポインタ）**

汎用レジスタなどのデータを退避するときに使用されるものです．1回の命令で2バイト分（つまり，16ビット・ペアレジスタとして）退避しますので便利です．

**▼PC（プログラム・カウンタ）**

次に実行する命令のメモリのアドレスが入っているところで，命令によって自動的にセットされます．

**▼IX，IY（インデックス・レジスタ）**

メモリを呼び出すための専用レジスタです．IX，IYは全く同じ働きをしています．

**▼R（メモリ・リフレッシュ・レジスタ）**

ダイナミック・メモリ（メモリのひとつで，多くのパソコンで使用）に使われているものです．乱数の基数などに利用されることがあります．

**▼I（割り込みベクトル）**

モード2という状態で使用されるベクトルで，このレジスタにはアドレスの上位バイトが入っています．しかし，今回は使用しません．

**▼IFF（インタラプト・レジスタまたはインタラプト・フリップ・フロップ）**

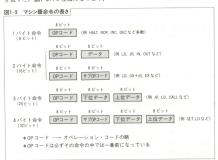
割り込み許可をするか，しないかを決めるもので，プログラム上で操作することができます．PC-8801mkIISRではCRTなどに使用されていますので，こ

のレジスタはプログラムによってコントロールする必要があります（割り込みについての説明は，P.241参照）．

CPUの内部構成を説明すると，以上のようになります．具体的な使用方法はプログラムを作りながら覚えていきますが，各レジスタの名前とビットの長さは今ここで覚えておいてください．

## マシン語命令の長さ

マシン語の命令の長さは一定ではなく，1バイト命令から4バイト命令まであります．特にハンド・アセンブルでアセンブリ言語をマシン語に変換するときには，このマシン語の命令の長さが必要となってきます．ハンド・アセンブルを行うときは，必ず対応表で調べてください．マシン語命令のバイトの長さを表すと，図1-5の5種類になります．

図1-5　マシン語命令の長さ

マシン語命令は，1バイト命令から4バイト命令まであります．Z-80A CPUは8ビット・マシンですから，1バイト命令を扱う際には問題はありません．しかし，4バイト命令などは一度にCPU内に取り込むことができません．そこでCPUは，命令を8ビット（1バイト）ずつ4回に分けて取り込むことになり

ます．CPUは命令の長さに応じて取り込む回数を自動的に調整するので，取り込む回数に注意する必要はありません．

## ■ハンド・アセンブルの際には上位と下位を逆に

図1-5の3バイト命令と4バイト命令を見てください．[下位データ][上位データ]と下位データの方が先に書いてありますね．これらの命令ではデータが2バイト・データなので当然CPUはデータの部分を2回に分けて取り込むことになります．このときCPUの癖が出て，2バイト・データの場合は必ず下位データから先に取り込みます．そこで，ハンド・アセンブルの際には注意が必要です．

## ■ JP 8800H をハンド・アセンブルすると…

たとえば，

8 8 0 0 H　（2バイト・データ）

上位データ　下位データ

JP　8800H　（アセンブリ言語で書いたジャンプ命令）

これをハンド・アセンブルしてみましょう．

正しいハンド・アセンブル作業

JP → C3
8800H → 00 88

＊上位バイト・データと下位バイト・データを逆にする

誤ったハンド・アセンブル作業

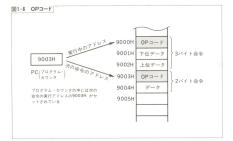
JP → C3
8800H → 88 00

以上のようになりますので，注意しましょう．なお，アセンブラを使用した場合にはアセンブラ自身で上位バイト・データと下位バイト・データを逆にしてくれますので注意する必要はありません（ただし，ダンプリストを見るときは上位と下位が逆になっていますので注意してください）．

## ■ OP コードは命令の頭にくる

OPコード（オペレーション・コード）は，必ずひとつの命令の中では一番前になっています．このOPコードを解読することによってCPUは，命令が何バイト命令であるかを理解するのです．

先程のCPUの内部構成の項でPC（プログラム・カウンタ）というものが出てきましたね．PC（プログラム・カウンタ）は次に実行するアドレスを示すものでした．次に実行するアドレスとは，次の命令のOPコードのあるアドレスのことです．たとえば現在の命令が3バイト命令であるならば，次のOPコードのあるアドレスは現在実行中のアドレスの3つ先ということになります(図1-6)．したがって，メモリの割り当てを間違えると，CPUは単なるデータをOPコードとして取り込みますので，暴走ということがおこります．

図1-6 OPコード

## Fレジスタを探る

コンピュータがすばらしいという理由のひとつに，ある条件に従って判断してくれるというのがあります．BASIC 言語では，「IF～THEN」などの命令によって判断を行っています．この判断のもとになっているのが，CPU の中にあるF（フラグ）レジスタです．マシン語を理解するには必ずこのF（フラグ）レジスタの意味を理解しなければなりません．

### ■演算結果の状態を表す旗

フラグ（Flag）とは旗のことです．よく運動会などで，赤旗と白旗を上げたり下げたりしてスタートの準備ができたかどうか信号を送っているのを見かけると思います．旗の上げ下げは準備状況を表しており，それを見て進行係は競技を進行させます．これと同じことが，CPU のフラグにもいえます．

フラグは，CPU が演算を行った結果こういう状態になった，ということを教えてくれるものなのです．

CPU のフラグとは，F（フラグ）レジスタの各ビットのことです．Fレジスタだけは他のレジスタと違って，データなどをロードしたりしません．CPU は演算命令を実行すると，ある種のデータをレジスタに書き込んでくれます．私たちはこのFレジスタのデータを見ることによって，演算結果の状態を知るこ

とができるわけですが，一般にマシン語のプログラムではFレジスタのデータを見る必要はありません．

分岐命令（たとえば，「JP Z」や「CALL C」など）ではフラグが立つ（フラグが1になること），立たない（フラグが0になること）ということをCPUが判断します．分岐命令はフラグが立ったときの命令と立たなかったときの命令とがペアになっているので，プログラムに合った命令を使用できます．

## ■Fレジスタの各ビットの意味

Fレジスタの各ビットの意味は図1-7のようになります．

図1-7　Fレジスタの各ビットの意味

### ▼ゼロ・フラグ

ある命令の実行結果，アキュムレータが0になったときにセット（1が立つ）されます．アキュムレータが0でないときはリセット（0になる）されます．よく使用され，同じ値であるか調べるときなども使用されます．

### ▼キャリー・フラグ（重要）

キャリー・フラグは次のような結果が生じた場合にセットされます．

・加算の結果，桁上りが生じた場合

・減算の結果，桁下りが生じた場合

・ローテート，シフトの場合

上記のような条件が立たなければ，キャリー・フラグはリセットされます．また，キャリー・フラグは演算の他にアキュムレータのビットを調べたりするのにも使用されます．

**▼サイン・フラグ**

符号付数字を扱う場合に使用され，演算結果が負（マイナス）の場合にセットされます．

**▼パリティ／オーバーフロー・フラグ**

このフラグは2つの機能を持っています．そのひとつは，論理演算を実行したときに，その結果アキュムレータ内で「1」を示すビットの個数が偶数ならばパリティ・フラグがセットされ（1が立つ），奇数ならばリセット（0になる）される機能です．

もうひとつは，符号付2の補数演算の結果オーバーフロー（桁あふれ）が生じたときにオーバーフロー・フラグがセットされる機能です．

**▼ハーフキャリー・フラグ**

アキュムレータの下位4ビット目から桁上りまたは桁下りがあったとき，セットされます．10進数の補正命令(DAA)を実行するときに，このフラグが用いられます．

**▼減算フラグ**

BCD（2進化10進）演算を補正するアルゴリズムは加算と減算とで異なります．そこで，DAA 演算が正しく補正処理を行えるようにこのフラグが用いられます．最後に実行された命令が減算ならセットされ(1になる)，加算ならリセット（0になる）されます．

以上，フラグ・レジスタの各ビットについてまとめてみました．この中で，特に覚える必要があるのは，

ゼロ・フラグ

キャリー・フラグ

の2つです．その他のフラグは，必要になったときに覚えてください．

*Section 3*

# PC-8801mkIISRのハードウェアを知ろう

## メモリはどう使われているか

BASICと違ってマシン語では，常にメモリの使い方を管理する必要があります．メモリマップ（メモリの使われ方）を理解していないと，マシン語のプログラムは作れません．本書では，メイン・メモリ以外のメモリも操作することになりますから，メモリマップをよく覚えておいてください．では，PC-8801mkIISRのメモリマップを**図1-8**に示します．

**図1-8　SRのメモリマップ**

〔メインバンク〕

| アドレス | メインバンク | | | | | | |
|---|---|---|---|---|---|---|---|
| FFFF–E600 | ワークエリア | G-VRAM 0（青） | G-VRAM 1（赤） | G-VRAM 2（緑） | | | |
| E600–C000 | 変数エリア | | | | | | |
| 83FF–8000 | テキストウィンドウ | | | | | | |
| 7FFF–6000 | テキストエリア | (N88モニタ) | N88-BASIC ROM | E0ROM | E1ROM | E2ROM | E3ROM |
| 6000–0000 | | N-BASIC ROM | | | | | |

（E0ROM～E3ROM：4th ROM）

### ■バンク切り替えで212Kバイトのメモリ空間

8ビット・マシンでは，一度に0000H番地からFFFFH番地までのメモリしか指定できません．すなわち，一度に持てるメモリ空間は64Kバイトまでということです．しかし，PC-8801mkIISRではバンク切り替えという特殊な方法を

使用しているので，212Kバイトのメモリ空間を持つことができます．もちろん，メイン・メモリ以外のメモリを指定するときはアドレスを2回に分けて指定しなければなりません．具体的な操作については，後章で練習しながら学んでいきましょう．特に，VRAM 0～2までのグラフィックス用のメモリは，グラフィックス画面やグラフィックスのカラー化に欠かせないものです．

なお，マシン語では主に，メイン・メモリとVRAM 0～2までを使用しています．

## システムからVRAMまでコントロールするI/Oアドレスマップ

前節ではメモリマップについて説明しましたが，ここでは，メモリのバンク切り替えを初めとして，すべての動作をコントロールするI/Oアドレスマップについて説明していきます．

CPU (Z-80A) は，メモリをコントロールする以外に特別に外部へ入出力するアドレスを00H～FFHまでの256番地持っています．PC-8801mkIISRでは，この外部へ入出力 (INPUT/OUTPUT) するアドレスをうまく使用して，メモリを増設させたりシステムをコントロールしたりするのです．したがって，マシン語ではI/Oアドレスマップを絶対に知る必要があります．I/Oアドレスマップを表1-4に示します．

表1-4　I/Oアドレスマップ

| I/Oアドレス | 機　能　内　容 |
|---|---|
| 00H～0BH | キーボード・マトリックスで，キーボードからの読み取りのときに使用される（第3章参照） |
| 10H | プリンタのDATAポート．プリンタへのデータがセットされる番地 |
| 20H，21H | カセットやRS-232Cインターフェースをコントロールする |
| 30H，31H，32H | システムポートで，システムをコントロールできる重要な番地 |
| 34H，35H | G-VRAM制御 |
| 40H | プリンタやRS-232Cのタイミングをコントロールする |
| 44H，45H | プログラマブル・サウンド・ジェネレータ(FM音源)の制御 |
| 50H～5FH | CRTコントロール．画面に関してすべてコントロールできる重要な番地 |

| | |
|---|---|
| 60H～68H | DMAコントロール（CPUを通さずに直接，メモリと周辺機器の間でデータを転送することをDMAと言い，CRTとVRAMの間で使用されている） |
| 70H，71H，78H | メモリコントロール．拡張ROMなどのコントロール |
| 80H～8FH<br>90H～9FH<br>A0H～AFH<br>B0H～BFH<br>C0H～CFH<br>D0H～DFH | 拡張用スロットバス．自由に使用できる |
| E2H，E4H，E6H | 主に割り込みコントロール．インタバルタイマーなどのコントロール |
| E8H～EBH | 漢字ROMコントロール |
| FCH～FFH | フロッピィディスクコントロール |

どうです？　わかりましたか？　これを見ただけでは，何をどのようにコントロールしているかわかりませんね．後章でプログラムを作りながら説明していきますので，そのときまでお待ちください．というのは，ここで説明してもプログラムがないと理解しにくいからです．ですから，ここではI/Oアドレスマップが何をコントロールしているかだけ理解してください．

## N88-DISK BASIC使用時のメモリマップ

マシン語のプログラムは，RAM 領域であればどこからでも作ることが可能です．しかし，実際にはどこからでもというわけにはいきません．その理由は N88-DISK BASIC を立ち上げたときには DOS(Disk Operating System;ディスクのファイル管理プログラム)が動作しているからです．この DOS をうまく使用しないと，ディスクにプログラムを“セーブ”，“ロード”できません．DOS をマシン語のプログラムで組むことも可能ですが，まず素人では無理でしょう．したがって，マシン語のプログラムエリアは事実上，DOS の部分のプログラムエリアを使用することはできません．

### ■マシン語プログラムエリアは B900H 番地から

そこで，DOS のエリアを壊さない範囲でマシン語のプログラムエリアを決定しなければなりません．DOS のエリアはオープンするファイル数や変数によってその都度変ってしまうので，安全性をとってファイル数を1にしたときのマシ

ン語プログラムエリアをB900H番地からとします(**図1-9**)．DOSのエリアを壊していないので，$N_{88}$-BASICコマンドの“BLOAD，BSAVE”が使用でき，モニタコマンドを使うよりずっと楽にマシン語プログラムのファイル管理ができます．

中途半端なB900H番地からマシン語プログラムエリアを始めている理由は，他にもあります．**図1-9**を見ればわかるように，グラフィックスのためのプレーン(G-VRAM)がC000H番地から始まっているために，プレーンを操作する関係上どうしてもC000H番地以前にマシン語プログラムエリアを設定したいからです．第4章(P.163)で説明してありますが，グラフィックスのプレーンを操作するためにはバンク切り替えという操作を必要とするので，C000H

**図1-9　$N_{88}$-DISK BASIC使用時のメモリマップ**

番地以降にバンク切り替えがなければなりません．

B900H 番地から E5FFH 番地までのメモリの大きさは10K バイト近くあります．これは，データを大量に扱うプログラムを作るのでなければ充分な大きさです．もし，これ以上の大きさが必要となったら，別のテキストエリアを使用すれば良いのですが，本書ではそこまで長いプログラムは作りません．

以上の理由から，特別の指示がない限り，マシン語プログラムエリアを

PC－8801mkIISR B900H～E5FFH番地

とします．

## ■モニタに入る前に

ただし，BASIC コマンドの「CLEAR 命令」でマシン語エリアの宣言をしていないので，BASIC モードに戻ったときに BASIC 命令が100％使用できません．もし，完全に使用したいのであれば，必ずモニタに入る前に次の命令を実行してください．

CLEAR，&HB8FF ↵

これで，すべての命令が使用できます．しかし，拡張命令のエリアとマシン語プログラムエリアが同じなので拡張命令（CMD のついた命令）は使用することができません．

# Section 4

# マシン語モニタを使いこなすために

Base

## 非常に強力なマシン語モニタ

マシン語の入力はBASICでもできますが，本格的に入力しようと思ったら，マシン語モニタのお世話にならなくてはなりません．PC-8801mkIISR用のモニタは，パソコン用としては非常に強力なものと思われます．特長は次のようになっています．

▼特長

①※ミニアセンブラ機能を持っています．
② 数値を16進数，8進数のどちらでも入力，表示が可能です．
③ I/Oポートへの入出力が可能です．
④※逆アセンブル機能を持っています．
⑤ CPUのレジスタの値の表示，変更が可能です．
⑥ カセット，フロッピィディスクにロード，セーブが可能です．
⑦ スクリーンエディタによるメモリ内容の変更ができます．
⑧ Helpコマンドにより，コマンドの入力形式がわかります．

（NECユーザーズマニュアルより）

### ■だけどまだまだ不満がある

これだけの特長を備えていながら，①と④については非常に不満です．なぜなら，このミニアセンブラはZ-80AというCPUのニーモニック（ザイログ社）を採用せず，Z-80Aの前の8080Aのニーモニック（インテル社）を採用しているからです．したがって，このミニアセンブラを使用する場合は，Z-80Aのアセンブリ言語は使用できず，8080Aのアセンブリ言語に直してからでないと入力できません．これでは，ニーモニックを2種類覚えなければならず，ユーザに負担がかかってしまいます．その上，Z-80Aが持っている強力な命令などは受け付けてくれないので，ハンド・アセンブルする必要があります．**表1-5**にZ-80Aと8080Aのニーモニックの対応表を掲げておきますので，このミニアセ

ンブラを使用したいときに利用してください．

次に④の逆アセンブル機能ですが，これも不満です．というのは，逆アセンブラとミニアセンブラとは対になっているからです．つまり，ミニアセンブラでは8080A アセンブリ言語しか使用できないということは，逆アセンブラでも8080A のアセンブリ言語しか使用できないことも意味するのです．

**表1-5　8080AとZ-80CPU（Z-80A）の命令対応表（8080Aにないものは省略）**

| 8080A ニーモニック | Z-80 ニーモニック | 8080A ニーモニック | Z-80 ニーモニック | 8080A ニーモニック | Z-80 ニーモニック |
|---|---|---|---|---|---|
| ACI | ADC A, N | IN | IN A,(N) | POP H | POP HL |
| ADC M | ADC A,(HL) | INR M | INC (HL) | POP PSW | POP AF |
| ADC r | ADC A, r | INR r | INC r | PUSH B | PUSH BC |
| ADD M | ADD A,(HL) | INX B | INC BC | PUSH D | PUSH DE |
| ADD r | ADD A, r | INX D | INC DE | PUSH H | PUSH HL |
| ADI | ADD A, N | INX H | INC HL | PUSH PSW | PUSH AF |
| ANA M | AND (HL) | INX SP | INC SP | RAL | RLA |
| ANA r | AND r | JC | JP C, NN | RAR | RRA |
| ANI | AND N | JM | JP M, NN | RC | RET C |
| CALL | CALL NN | JMP | JP NN | RET | RET |
| CC | CALL C, NN | JNC | JP NC, NN | RLC | RLCA |
| CM | CALL M, NN | JNZ | JP NZ, NN | RM | RET M |
| CMA | CPL | JP | JP P, NN | RNC | RET NC |
| CMC | CCF | JPE | JP PE, NN | RNZ | RET NZ |
| CMP M | CP (HL) | JPO | JP PO, NN | RP | RET P |
| CMP r | CP r | JZ | JP Z, NN | RPE | RET PE |
| CNC | CALL NC, NN | LDA | LD A,(NN) | RPO | RET PO |
| CNZ | CALL NZ, NN | LDAX B | LD A,(BC) | RRC | RRCA |
| CP | CALL P, NN | LDAX D | LD A,(DE) | RST | RST P |
| CP E | CALL PE, NN | LHLD | LD HL,(NN) | RZ | RET Z |
| CPI | CP N | LXI B | LD BC, NN | SBB M | SBC A,(HL) |
| CPO | CALL PO, NN | LXI D | LD DE, NN | SBB r | SBC A, r |
| CZ | CALL Z, NN | LXI H | LD HL, NN | SBI | SBC A, N |
| DAA | DAA | LXI SP | LD SP, NN | SHLD | LD (NN), HL |
| DAD B | ADD HL, BC | MVI M | LD (HL), N | SPHL | LD SP, HL |
| DAD D | ADD HL, DE | MVI r | LD r, N | STA | LD (NN), A |
| DAD H | ADD HL, HL | MOV M, r | LD (HL), r | STAX B | LD (BC), A |
| DAD SP | ADD HL, SP | MOV r, M | LD r, (HL) | STAX D | LD (DE), A |
| DCR M | DEC (HL) | MOV r1, r2 | LD r, r | STC | SCF |
| DCR r | DEC r | NOP | NOP | SUB M | SUB (HL) |
| DEX B | DEC BC | ORA M | OR (HL) | SUB r | SUB r |
| DCX D | DEC DE | ORA r | OR r | SUI | SUB N |
| DCX H | DEC HL | ORI | OR N | XCHG | EX DE, HL |
| DCX SP | DEC SP | OUT | OUT (N), A | XRA M | XOR (HL) |
| DI | DI | PCHL | JP (HL) | XRA r | XOR r |
| EI | EI | POP B | POP BC | XRI | XOR N |
| HLT | HALT | POP D | POP DE | XTHL | EX (SP), HL |

注．$N_{88}$-BASICモニタを使用する場合は，Z-80ニーモニックを8080Aニーモニックに変えて使用します．

**表1-6**

| Z-80A | モニタのニーモニック |
|---|---|
| JR　　〈アドレス〉注1 | JMPR〈アドレス〉注2 |
| JR Z,〈アドレス〉 | JRZ　〈アドレス〉 |
| JR NZ,〈アドレス〉 | JRNZ〈アドレス〉 |
| JR C,〈アドレス〉 | JRC　〈アドレス〉 |
| JR NC,〈アドレス〉 | JRNC〈アドレス〉 |
| DJNZ〈アドレス〉 | DJNZ〈アドレス〉 |

注1　Z-80Aの相対ジャンプ(注3)のアドレスは符号付2の補数を指します．
注2　モニタのアドレスは絶対番地（A000H番地など）で指定します．モニタの中でこれを符号付2の補数に計算してくれます．
注3　相対ジャンプについては後章で説明します．

逆アセンブラとは，マシン語（2進数もしくは16進数で書かれたもの）で書かれたものをアセンブリ言語に直してくれるものです．使い方は，デバッグのときにエラーが出た箇所のマシン語を逆アセンブルして，初めの命令通りアセンブルされているかどうかを調べます．その他に，他人の書いたマシン語を解析するとき16進数で書かれているものは解析しにくいので，逆アセンブルにかけてアセンブリ言語に直してから解析すれば非常に理解しやすくなります．

### ■市販のアセンブラ（DUAD-88D）を使う

8080Aのニーモニックでは，Z-80Aの能力を100%生かしきれません．したがって，もし100% CPUの能力を生かしたいのであれば，このミニアセンブラを使用せずすべてハンド・アセンブルするか，Z-80A用のアセンブラを使用するしかありません．本書のリストはすべて，DUAD-88D（アスキー）というZ-80Aのアセンブラを用いて作成しています．もし，本格的にマシン語を覚えたいのであれば，高価（49,800円）ですが損はないでしょう．

なお，PC-8801mkIISRでDUAD-88Dを使用する際は，BASICモードスイッチをN88V1にセットしてください．

NECとマイクロソフトが，どんな理由で8080Aのニーモニックをミニアセンブル機能に用いたのかはわかりません．次期新製品はユーザのためにも，CPUにあったニーモニックを採用していただけたらと思います．

## いよいよマシン語の世界へ

まず，BASICモードとマシン語モニタのプロンプトの違いについて説明しておきましょう．プロンプトの違いは，このようになります．

**OK……………………BASICが扱える状態**
**h］または q］……マシン語が扱える状態**

プロンプトは入力モードであることをユーザに表示するものです．したがって，どのプロンプトが表示されているかによって，どのモードであるかユーザが判断することになります．

## ■モニタを起動する

次にマシン語モニタの起動とBASICモードへ戻る方法について説明しましょう．

### ① BASICモードからマシン語モニタを起動する

### ②マシン語モニタからBASICモードへ戻る方法

# マシン語モニタ

マシン語のコマンドは全部で22種類あり，使い方によって非常に便利なものもあります．モニタコマンドの種類をまとめると，次のページの**表1-7**のようになります．

## ■モニタ使用にあたって

マシン語モニタを使用するにあたっては，多少の注意が必要です．ユーザーズ・マニュアルにはカセットテープについての規則も挙げてありますが，ディスク装置がついているのにわざわざカセットテープを使用することもないので，ここでは挙げません．ここでは使用上の規則として，数値入力，計算機能，画面形式の３つを挙げておきます．

表1-7 マシン語モニタ コマンド

| コマンド | 意　　味 | 機　　能 |
|---|---|---|
| A | アセンブル | 入力した1行をアセンブルします. |
| B | ベース | 数値の表現形式を変えます.(8進↔16進) |
| D | ダンプ・メモリ | メモリの内容をディスプレイに表示します. |
| E | エディット・メモリ | スクリーン・エディタの機能を用いてメモリの内容を変更します. |
| F | フィル・メモリ | メモリの内容を定数で埋めていきます. |
| G | ゴー | ユーザプログラムを実行します. |
| I | インプット | I/Oポートの値を読み込みます. |
| L | ディス・アセンブル | 機械語を逆アセンブルします. |
| M | ムーブ・メモリ | ある範囲のメモリの内容を他のアドレスのメモリ領域へ移します. |
| O | アウトプット | I/Oポートへデータを出力します. |
| P | プリンタ・スイッチ | プリンタへの出力をコントロールします. |
| R | リード・テープ | カセットテープからデータをロードします. |
| S | セット・メモリ | メモリにデータをセットします. |
| TM | テスト・メモリ | メモリをテストします. |
| V | ベリファイ・テープ | カセットテープの内容と,メモリの内容を比較します. |
| W | ライト・テープ | メモリの内容をカセットテープにセーブします. |
| X | イグザミン・レジスタ | CPUのレジスタの値を調べ,変更します. |
| HELP または CTRL-A | ヘルプ | コマンドとそのパラメータの形式をディスプレイに表示します. |
| CTRL-B | リターン | BASICモードへ復帰します. |
| CTRL-D * | ダンプ・ディスク | フロッピィディスクの内容をディスプレイに表示します. |
| CTRL-R * | リード・ディスク | フロッピィディスクから,データをロードします. |
| CTRL-W * | ライト・ディスク | フロッピィディスクへデータをセーブします. |

* がついてる3つのコマンドはN-BASICでは使えません.

## ■数値は16進数で入力

PC-8801mkIISR は，16進数と8進数の両方を扱うことができますが，現在では16進数が定着し，8進数を扱うことはあまり多くありません．と言うよりは，ほとんど使用しません．もし扱うとすれば，CPU の OP コードの解析に多少便利なくらいです．それな程度ですから，8進数のモードは必要ないでしょう．

・16進数モード

| | |
|---|---|
| アドレス | 4桁：0000H〜FFFFH |
| データ | 2桁：00H〜FFH |

参考

・8進数モード

| | |
|---|---|
| アドレス | 6桁：000000O〜777777O |
| データ | 3桁：000O〜377O |

注…Oは8進数を表す記号（Octal）

アドレスもデータも最後から4桁，2桁が有効になりますので，途中で入力ミスをした場合はその時点から正しいアドレス4桁なり，データ2桁なりを再入力できます．もちろん，16進数の記号であるHはつける必要がありません．

## ■便利な計算機能

コンピュータでは16進数を利用しているために，暗算でたし算や引き算などができません．たとえばアドレスの計算などは一度10進数に変換して計算し，また16進数に直すということになります．BASIC のコマンドは16進数を扱えますので利用できますが,BASIC モードにしなければならないので不便です．しかし，モニタの各コマンドレベルで加減（ただし，かっこや累乗は使用できません）ができますので，アドレスなどの計算に威力を発揮してくれます．

例．F3F0H 番地に50H をたした結果のアドレスを求めた場合

```
h] SF3F0+50
F440  00
└→求めたアドレス
```

## ■モニタに入る前に“WIDTH　80，25”

マシン語モニタに入るときは，BASIC の“WIDTH”命令がそのまま引き継がれますので，画面モードを決めるには，モニタに入る前に

WIDTH　80，25

と命令しておいた方が便利です．マシン語でも画面モードを設定できますが，BASIC で行った方が楽でしょう．

のちに説明するEコマンドは，テキスト画面のスクロールウィンドウの影響を受けることがありますので，モニタに入る前に次の命令を実行して下さい．

CONSOLE 0,（17以上の値）

# 各コマンドはこう使う

## ■ A コマンド（Assemble＝アセンブル）

入力形式　　A　格納開始番地

例　h］AD000 ↵

格納開始番地（ここでは D000H 番地）

このAコマンドはミニアセンブラで，インテル8080ニーモニックで入力した1行をアセンブルし，できたマシン語をメモリに格納します．もし，Aコマンドを使用するときに格納番地を指定しない場合は，前に実行したAコマンドの次の番地となります．実行してみましょう．

```
h]AD000
D000                ■
                    ↑
                カーソルの位置
```

```
h]AD000
D000                MVI  A,30■
```

▼画面 1

Aコマンドで格納開始番地をD000Hと指定して↵キーを押すと，D000H 番地が表示され，カーソルは15個のスペースを空けた位置で，ニーモニックの入力待ちになります．

▼画面 2

ニーモニックを入力（Z-80用では，LD　A，30H）します．30は16進数なのですが，プロンプト（h］）が16進数入力となってますので，16進数を示す“H”はつけていません．

▼画面 3

ニーモニックを入力して↵キーを押すと，スペースを空けておいた同じ行にアセンブルされたマシン語が表示され，即，指定され

```
h]AD000
D000  3E 30          MVI  A.30
D002  D3 30          OUT  30
D004  00             NOP
D005  00             NOP
D006  76             HLT
D007                 ■
```

た番地からメモリに格納します．このニーモニックは２バイト命令であることがわかります．次の入力待ちになります．

**▼画面４**

このマシン語のプログラムは意味のないものですが，このような画面になります．［INS DEL］キーが使用できますから，間違えて入力した場合などは入力した文字を訂正できます．

**▼モニタに戻るには**

Ａコマンドのアセンブラからモニタへ戻るには，次の３つの方法が使用できます．

① ［STOP］キーを入力する
② ［CTRL］キーと［C］キーを同時に入力する
③ アドレスが表示されてニーモニックの入力待ちのときに［↵］キーを入力する（画面１のときや，画面４の D007H 番地が表示されているとき）

## ■Ｂコマンド（Base＝ベース）

**入力形式**　　h] B　QまたはH

例　h]　h] BQ ［↵］
　　　　　　└──入力する数値を８進数にする

数値をキーボードから入力したり，CRT に表示する場合に，16進数扱いにするか，８進数(Octal)扱いにするかを設定します．$N_{88}$-BASIC からモニタへ切り替えたときは，16進数モードになっています．８進数は現在のところ，あまり使い道がないようです．プロンプトの違いは，

16進数 → h]
８進数 → q]

となります．では，さっそく実行してみましょう．

```
h]BQ
q]■
```

**▼画面１**

16進数モードになっているのを，“BQ”を入力して８進数モードに切り替えた状態です．

```
q]BH
h]■
```

**▼画面 2**

8進数モードになっているものを“BH”を入力して16進数モードに切り替えた状態です．

## ■Dコマンド（Dump＝ダンプ）

**入力形式**　　D　表示開始番地，表示終了番地

例1　h] D0000, 00FF ↵　(0000H 番地から00FFH までのメモリの内容を表示します)

例2　h] D0000 ↵　(0000H 番地から16バイト分表示します)

例3　h] D, 03FF ↵　(前に実行されたDコマンドの最後の番地から，03FFH 番地までの内容を表示します)

例4　h] D ↵　(前に実行されたDコマンドの最後の番地から16バイト分の内容を表示します)

Dコマンドはメモリの内容をCRTに表示するコマンドで，マシン語のプログラムやデータなどが入っていますので，これらを参照するときに使用します．このコマンドで便利なのは対応するアスキー文字(正確にはPC-8801mkIISRのキャラクタ)を右側に表示してくれますので，マシン語などのプログラムの中にあるメッセージ等を探す場合に利用できることです（ただし，16進数モードのみ)．では，実行してみましょう．

```
h]D0000.00FF
0000 F3 31 A0 E1 C3 E5 3B 00 7E E3 BE 23 E3 C2 93 03   ﾖ1 ﾄﾃ◣; ~ｦｾ#ｦｯﾄ
0010 23 7E FE 3A D0 C3 15 0A F5 CD 42 ED C3 25 59 00   #~ :ﾐﾃ  時ﾍB○ﾃ%Y
0020 7C 92 C0 7D 93 C9 00 00 3A 44 EC B7 C2 83 20 C9   |ｲﾀ}ﾄﾉ  :D●ｷﾂ▬ ﾉ
0030 3A BD EA FE 08 C3 27 15 C3 69 E6 00 00 00 00 00   :ｽ◆  ﾃ'  ﾃi◥
0040 A0 21 FD 21 7E A7 C8 FE 20 C9 21 FF FF 22 56 E6    ! !~ｧﾈ  ﾉ!  "V◥
0050 23 C9 22 41 EC C1 C1 4F 78 D3 71 79 C1 C3 E9 11   #ﾉ"A●ﾁﾁOxﾓqyﾁﾃ♥
0060 CD BD 05 CD 21 4F C3 D6 4E CD 9F ED F3 3A C2 E6   ﾍｽ ﾍ!OﾃﾖNﾍ ○月:ﾂ◥
0070 C9 00 3A B9 E6 A7 37 C4 21 40 C3 6A 6F CD 96 18   ﾉ :ｹ◥ｧ7ﾄ!@ﾃjoﾍ|
0080 F5 7A FE 02 20 01 15 F1 C9 00 79 79 7C 7C 7F 50   時z      円ﾉ yy|| P
0090 46 3C 32 28 7A 7B 3E 22 00 00 A0 21 56 22 14 22   F<2(z{>"   !V" "
00A0 24 24 1D 24 53 25 29 26 99 21 E9 1D E6 1D 53 1F   $$ $S%)&┐!♥ ◥ S
00B0 B7 1F 34 21 3A 23 2F 23 5A 23 41 13 5F 21 D6 00   ｷ 4!:#/#Z#A _!ﾖ
00C0 6F 7C DE 00 67 78 DE 00 47 3E 00 C9 00 00 00 35   o|ﾞ gxﾞ G> ﾉ   5
00D0 4A CA 99 39 1C 76 98 22 95 B3 98 0A DD 47 98 53   Jﾊ┐9 v┌"━ｳ┌ ﾝG┌S
00E0 D1 99 99 0A 1A 9F 98 65 BC CD 98 D6 77 3E 98 52   ﾑ┐┐  ╯┌eｼﾍ┌ﾖw>┌R
00F0 C7 4F 80 06 0B 06 0B 06 0B 06 0B 06 0B 06 0B 06   ﾇO_
```

**▼画面 1**

$N_{88}$-BASICのROMの表示

```
h]D0160.02D5
0160 FF FF 01 00 00 01 81 00 01 FF FF 00 06 36 01 13        -     6
0170 01 19 00 20 00 00 FF 00 00 00 00 00 00 00 22 11              "
0180 00 FF C8 F3 00 00 00 00 00 CD EF 00 00 1C 00 00     ｽ月    ヘ\
0190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01A0 00 00 00 00 00 00 00 00 00 00 FF 00 15 CB EE 00            ヒ/
01B0 6C 6F 61 64 20 22 00 00 00 00 00 00 00 00 00 00    load "
01C0 61 75 74 6F 20 00 00 00 00 00 00 00 00 00 00 00    auto
01D0 67 6F 20 74 6F 20 00 00 00 00 00 00 00 00 00 00    go to
01E0 6C 69 73 74 20 00 00 00 00 00 00 00 00 00 00 00    list
01F0 72 75 6E 0D 00 00 00 00 00 00 00 00 00 00 00 00    run
0200 73 61 76 65 20 22 00 00 00 00 00 00 00 00 00 00    save "
0210 6B 65 79 20 00 00 00 00 00 00 00 00 00 00 00 00    key
0220 70 72 69 6E 74 20 00 00 00 00 00 00 00 00 00 00    print
0230 65 64 69 74 20 2E 0D 00 00 00 00 00 00 00 00 00    edit .
0240 63 6F 6E 74 0D 00 00 00 00 00 00 00 00 00 00 00    cont
0250 6C 69 74 65 72 61 6C 00 00 00 00 00 00 00 00 00    literal
0260 68 61 6C 66 2F 66 75 6C 6C 00 00 00 00 00 00 00    half/full
0270 4C 50 54 20 65 6E 61 62 6C 65 00 00 00 00 00 00    LPT enable
0280 63 6F 70 79 20 62 75 66 66 65 72 00 00 00 00 00    copy buffer
0290 4C 50 54 20 66 65 65 64 00 00 00 00 00 00 00 00    LPT feed
02A0 C3 00 00 C3 00 00 FF E5 E5 21 67 31 F5 3A C2 E6    テ  テ   ◣◣!g1時:ツ◥
02B0 F5 E6 FB D3 31 32 C2 E6 CD 91 33 F3 F1 D3 31 32    時◥ モ12ツ◥ヘ┬3月円モ12
02C0 C2 E6 F1 E1 FB C9 E5 21 80 30 18 E0 E5 21 43 41    ツ◥円ﾄ ◢◣!_0 ═◣!CA
02D0 18 DA E5 21 79 30                                  レ◣!y0
```

**▼画面 2**

N₈₈-BASIC でのファンクションキーの指定内容は，01B0H 番地から0244H 番地までで指定してあります．しかし，ROM の中に指定してあるので変更できません．

**▼データをチェックしたいときなど**

Pコマンド（P.49参照）でプリンタ・スイッチがプリント・モードに指定されていれば，ダンプされた結果がプリンタへも出力されます．ただし，入力形式が表示されないので，画面内に入るものだったら COPY キーを使用した方が良いでしょう．

また，Dコマンドを実行中に，画面表示を一時停止してデータをチェックしたいときなどは次のようにします．

画面上の表示を一時停止させる…………［CTRL］＋［S］
表示を再開する……………………………［STOP］，［CTRL］＋［C］以外のキー
モニタの入力モードに戻る…………………［STOP］，［CTRL］＋［C］

## ■Eコマンド（Edit＝エディット）

［入］［力］［形］［式］　　　E　開始番地

例　h］ED000 ⏎
　　└────エディットの開始番地を D000H 番地とする

Eコマンドは，スクリーンエディタの機能を利用して，CRTに表示されたメモリの内容を変更するコマンドです．このEコマンドは開始番地を指定すると自動的に256バイトの内容を表示し，さらに左側にはその1バイトに対応するアスキー文字を表示します．

**▼メモリ内容を変更するには**

メモリ内容を変更する際には，修正したいデータのところまでカーソル移動キー（↑↓←→）やロールアップ，ロールダウンキー（ROLL UP，ROLL DOWN）を使用してカーソルを移動させ，新しいデータを入力します．新しいデータが入力されると，メモリの内容は変更され，左側に表示されているアスキー文字も変更になります．このEコマンドは，マシン語のプログラムを入力したり，一部変更したりするのに便利です．

なお，使用できる編集キーは，

↑↓←→，ROLL UP，ROLL DOWN

で，モニタへ戻るときは，

STOP または ESC を押します．

```
h]ED000   ─カーソルの位置
D000 ■0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

**▼画面**

ED000⏎をしたときの画面で，カーソルはD000H番地に来ています．

## ■Fコマンド（fill＝フィルメモリ）

**入力形式**　　F　開始番地，終了番地，定数

例　h] FD000,　D0FF,　00 ⏎
　　　開始番地　終了番地　セットする数値

Fコマンドは指定した番地から番地まで，指定した数値をセットするコマンドです．このコマンドは，メモリの一部をクリアしたり，メモリの一部をある定数にセットしておき暴走をくい止めたりするときなどに使用します．

```
h]FD000.D0FF.FF
h]DD000.D0FF
D000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D010 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D020 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D050 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D070 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D080 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D090 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D0A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D0B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D0C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D0D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D0E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
D0F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

**▼画面**

D000H番地からD0FFH番地までFFHをセットし，セットされたかどうかDコマンドを使用して確認しています．

## ■Gコマンド（Go＝ゴー）

**入力形式**　G　実行開始番地，ブレイク・ポイント・アドレス#1，ブレイク・ポイント・アドレス#2

例1　h］GD000，D007 ⏎（ブレイク・ポイントが1つのとき）
例2　h］GD000，D007，D010 ⏎（ブレイク・ポイントが2つのとき）
例3　h］GD000 ⏎

GコマンドはマシGン語のプログラムを実行させるコマンドで，BASICで言えばRUNに相当するものです．このGコマンドは入力形式に記述してあるように，ブレイク・ポイントを2つまで設定することができます．

**▼ブレイク・ポイントとは**

ブレイク・ポイントとは，マシン語プログラム実行中に指定した番地でストップしたい場合に設定するもので，マシン語プログラムをデバッグするときにこの番地（ブレイク・ポイント）を通過するかしないかを確認するものです．ですから，通過する予定の番地を通過しなかった場合は，それ以前の番地にバグがあるということになります．

なお，ブレイク・ポイントが2つまで設定できるので，サブルーチンがあるときに一方はメイン・プログラムに設定し，もう一方はサブルーチンに設定して流れがどう変ったかなどを調べることができます．

ブレイク・ポイントを設定すると，そこを通過した場合，ブレイク・ポイントの番地で実行を中止してモニタに戻ってしまいます．このときCPUのレジスタの値は保存されていますので，Xコマンド(P.53参照)でCPUのレジスタを見ることができます．ここでPC(プログラム・カウンタ)レジスタを見れば，設定したブレイク・ポイントであるかどうか知ることができます．

```
D000   00        NOP
D001   00        NOP
D002   00        NOP
D003   00        NOP
D004   00        NOP
D005   00        NOP
D006   00        NOP
D007   00        NOP
D008   00        NOP
D009   00        NOP
D00A   00        NOP
D00B   00        NOP
D00C   00        NOP
D00D   00        NOP
D00E   00        NOP
D00F   FF        RST   7
```

**▼画面 1**

NOP命令は何もしない命令で，ただ通過するだけのものです．RST 7はモニタへ戻るための命令です(88に限る)．したがってこれは，D000H番地からプログラムを実行すると，何もしないでD00EH番地まで実行し，D00FH番地を実行するモニタへ戻るプログラムです．

```
h]GD000
h]X
A :00 F :PZ---E-- B :0000 D :EDCC H :0001 A':00 F':PZ---E-- B':0000 D':FF7B
H':0911 IX:2080 IY:7DC9 I :F3 PC:D00F SP:E5F9
```

**▼画面 2**

ブレイク・ポイントを設定しないときは，PCレジスタはマシン語のプログラムを実行した最後のアドレスになっています．

```
h]GD000.D007
h]X
A :00 F :PZ---E-- B :0000 D :EDCC H :0001 A':00 F':PZ---E-- B':0000 D':FF7B
H':0911 IX:2080 IY:7DC9 I :F3 PC:D007 SP:E5F9
```

**▼画面 3**

ブレイク・ポイント・アドレスをD007H番地とした場合，プログラムがブレイク・ポイントのD007H番地でストップしたことがPCを見るとわかります．

```
h]GD000.D00A
h]X
A :00 F :PZ---E-- B :0000 D :EDCC H :0001 A':00 F':PZ---E-- B':0000 D':FF7B
H':0911 IX:2080 IY:7DC9 I :F3 PC:D00A SP:E5F9
```

**▼画面 4**

ブレイク・ポイント・アドレスをD00AH番地に設定しプログラムを実行させたとき，やはり，ブレイク・ポイントのD00AH番地でストップしたことがわかります．

注意

ブレイク・ポイントは，必ずRAMエリアにあり，プログラムが多分通過するであろう番地に設定しなければ効果がありません．

## ■Iコマンド（Input＝インプット）

入力形式　　　　I ポート・アドレス

例　h］ I 80 ⏎　(I/Oの80ポートから1バイトのデータを読む)

Iコマンドは，直接ポート・アドレスから1バイトのデータを読み込んで，CRTに表示するコマンドです．自作したI/Oポートをテストするときなどに使用します．プログラムを組まずにI/Oポートのデータを読めるので，テストが楽になります．

```
h]I70
00
h]O70,20
h]I70
20
h]
```

**▼画面**

これはI/Oポートの70Hを読み出したものです．00Hというデータが表示されています．これは，テキストウィンドウ(P.255参照)の上位バイトのデータです．テキストウィンドウの上位バイトはOコマンドで設定できるので，ためしに20Hに設定します．

なお，テストが終ったならOコマンドで必ず00Hに戻しておいてください．それには，次のようにします．

h］ O 70，00 ⏎

## ■Ｌコマンド（Disassemble＝ディスアセンブル）

入力形式　　L　開始番地，終了番地

例１　h］L0000，001F ↵　(0000H 番地から001FH 番地まで，逆アセンブルします)

例２　h］L0000 ↵　(終了番地を省略した場合は開始番地から＋15～17バイト分逆アセンブルします)

例３　h］L，002F ↵　(開始番地を省略した場合は，逆アセンブルの終了した番地から，指定した番地まで逆アセンブルします)

Ｌコマンドは逆アセンブラで，マシン語のプログラムをインテル社の8080ニーモニックに直すものです．この逆アセンブラは，マシン語のプログラムをデバッグするのに使います．すなわち，マシン語のプログラムを逆アセンブルしてみて，もとのアセンブリ言語と同じであるかどうかを見るのです．もし，同じにならなかったならば，ハンド・アセンブルが間違っていることになります．

**▼解析にも使える**

また，Ｌコマンドはマシン語プログラムの解析にも用いられます．解析するのに16進数はわかりにくいですから，逆アセンブラにかけてアセンブリ言語に直してから解析を行います．ただし，Ｄコマンドを使用して文字列を除いてからでないとうまくいきません（インテル社の8080ニーモニックだけを使用していればですが…）．

```
h]L0000.001F
0000   F3          DI
0001   31 E1A0     LXI   SP.E1A0
0004   C3 3BE5     JMP   3BE5
0007   00          NOP
0008   7E          MOV   A.M
0009   E3          XTHL
000A   BE          CMP   M
000B   23          INX   H
000C   E3          XTHL
000D   C2 0393     JNZ   0393
0010   23          INX   H
0011   7E          MOV   A.M
0012   FE 3A       CPI   3A
0014   D0          RNC
0015   C3 0A15     JMP   0A15
0018   F5          PUSH  PSW
0019   CD ED42     CALL  ED42
001C   C3 5925     JMP   5925
001F   00          NOP
```

**▼画面**

$N_{88}$－BASIC の ROM の内容をディスアセンブルしたときの例です．割り込み（P.241参照）を禁止してから，スタック・エリアを決めて3BE5H番地へジャンプしていることがわかります．このように，Ｌコマンドは解析もできます．

また，Lコマンド実行中に実行を一時中断したいときなどは，次のようにします．

画面の表示を一度停止させる…………[CTRL]+[S]
再画面を表示させる……………………[STOP]，[CTRL]+[C]以外のキー
モニタへ戻る……………………………[STOP]，[CTRL]+[C]

## ■Mコマンド（Move＝ムーブ・メモリ）

**[入][力][形][式]**　　M　転送するメモリの先頭番地，転送するメモリの最終番地，転送先の先頭番地

例　h］MD000，D00F，D100 ↵（D000H番地からD00FH番地までのデータをD100Hからにコピー）

Mコマンドは，指定された番地のメモリのデータを指定する番地へ転送するものです．このコマンドの使い方は3つあります．第1に，相対番地のプログラムを作ったときのテストに使えます．第2に，使用しているメモリのエリアを他のプログラムが使用する場合，エリアのメモリ内容を一時退避するのに使います．第3に，マシン語のプログラムを実行するときに予備としてそのプログラムをとっておくのに使用できます．

```
h]SD000
D000 00-00 00-01 00-02 00-03 00-04 00-05 00-06 00-07 00-08 00-09 00-0A 00-0B
D00C 00-0C 00-0D 00-0E 00-0F 00-
h]DD000
D000 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
h]DD100
D000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
h]MD000.D00F.D100
h]DD100
D100 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
```

**▼画面**

①Sコマンドを使用して，D000H番地からD00FH番地までに00H～0FHまでのデータを書き込みます．
②Dコマンドで，データが書き込まれているかどうかを確かめます．
③これから転送するD100H～D10FH番地までのデータを調べます．これで，D000H～D00FH番地までのデータとは違うことがわかります．
④Mコマンドを使用して，D000H～D00FH番地までのデータをD100H番地から転送します．

⑤Dコマンドを使ってデータがちゃんと転送されたか確かめます．転送されていることがわかります．

## ■Oコマンド（Output＝アウトプット）

入力形式　　　O　ポートアドレス，データ

例　h］O 30，C0 ⏎（I/O ポート30H に C0H というデータを送ります）

Oコマンドは，指定したポートアドレスにデータを出力します．主に自作したI/Oポートをテストするのに使います．直接I/Oポートにデータを出力できるので，テスト時間が短くなり非常に便利です．PC-8801mkIISR は I/O ポートを利用してマシンのシステムを作っていますので，このコマンドを使用してイタズラができます．システム全体がわかれば，イタズラではなくてシステムのコントロールができます．たとえば，モノクロ画面からカラー画面に変えることなどもできます．

```
h]I30 ──①
C3
h]O30,C0 ──②
h]O30,C3 ──③
```

▼画面

①I/O ポートの30H のデータが何であるか調べます．すると，C3H というデータであることがわかります．

②Oコマンドを使用して I/O ポートの30H に C0H というデータを出力してみましょう．その結果どうなるでしょうか？　実行してみてください．多分，画面にカラーの“＝”が表示されるはずです．ここでは，画面のモードに関するコントロールをしていることがわかります．

③このままでは画面の文字が読みとれないので，Oコマンドを使用して，元のデータを出力してあげます．すると，元のモノクロ画面になります．

## ■Pコマンド（Printer sw＝プリンタ・スイッチ）

入力形式　　　P

例　h］P ⏎

PコマンドはDコマンド，Lコマンド，CTRL-D コマンドを実行したときにプリンタに出力するか，しないかを決定するものです．これは画面に出力されるものと同じものをプリンタに出力するものですが，入力したコマンドまではプリンタに出力されませんので，それも出力したいときは COPY キーを使った方が良いかもしれません．

また，プリンタが ON のときと OFF のときではプロンプトが違いますので，プリンタに出力されるかどうかすぐにわかります．プロンプトの違いは**表1-8**の

ようになります．

表1-8　プロンプトの違い

| | プリンタ　ON | プリンタ　OFF |
|---|---|---|
| プロンプト | h)<br>q) | h]<br>q] |

Pキーを1回押すとプリンタがONに，もう1度押すとOFFになります．

```
h]P
h)■
h)P
h]■
```

▼画面

1回Pを押すと，h）が現れてプリンタがONになったことがわかります．もう1度Pを押すと，h]が現れてOFFになったことがわかります．

## ■Rコマンド（Read tape＝リード・テープ）

入力形式　　R　ファイル名

例1　h］R test↵（testというファイルをサーチしてみつけたら，ロードします）

例2　h］R↵（最初に見つけたファイルをロードします）

注……ファイル名は6文字以内です．6文字を越える場合は，最初の6文字のみが有効となります．

Rコマンドは，カセットテープからデータをロードするコマンドです．ファイル名を指定しない場合は最初に見つけたファイルをロードし，ファイル名を指定した場合は指定されたファイルを見つけるまでサーチします．

```
h]RTEST ――――― ①
Found:TEST

h]R ―――――――― ②
h]DD000
```

▼画面

①ファイル名を指定したときは，このようにサーチしたファイル名を表示しロードします．

②ファイル名を指定しなかったときは，最初に見つけたファイルをロードします．

## ■Sコマンド（Set memory＝セット・メモリ）

入力形式　　S　開始番地

例　h］SA000↵

Sコマンドは，メモリのデータを書き替えるコマンドです．データを変更し

ないでデータの確認をするときにも利用できて便利です．

Sコマンドを実行すると指定した番地が表示され，次にそのときのデータが表示されてデータの入力待ちとなります．ここでデータを変更する場合は，数値データを入力しスペースキーを押します．変更しないときはスペースキーを押します．すると，データは前のままで次の番地へ進み，次の番地のデータを表示して入力待ちになります．もし，Sコマンドを終了したい場合には，リターンキーを押します．データをセットした後にリターンキーを押すと，最後のデータはメモリにセットされます．

なお，数値データは16進数は最後の2桁，8進数は最後の3桁が有効です．

```
h]SD000
D000 00-00 00-01 00-02 00-03 00-04 00-05 00-06 00-07 00-08 00-09 00-0A 00-0B
D00C 00-0C 00-0D 00-0E 00-0F 00-
h]DD000
D000 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
```

**▼画面**

A000H 番地から00H～0FH までのデータを入力し，Dコマンドで確認．

また，メモリの番地を1つ進めたり，戻したりといった細かい動作は次のようにします．

- メモリの番地を1つ進める………………スペース・キー
- メモリの番地を1つ戻す…………………[CTRL]＋[B]，[←]
- メモリへ戻る………………………………[RETURN]

## ■ TMコマンド（Test Memory＝テスト・メモリ）

[入][力][形][式]　　TM

例　h] TM⏎

このコマンドはあまり使用することのないものですが，定期的に使用することによって，マシンの高信頼性を得ることができます．TMコマンドを実行すると，本体に内蔵されているRAMがすべてチェックされます．

**▼不幸にもBEEP音が鳴ったら…**

TMコマンドが実行されると，CRT上にいろいろな文字が表示されますが，気にする必要はありません．もし，不幸にもBeep音が鳴り続いたら，CRTを見てください．CRT上に不良メモリのアドレスが表示されるはずです．こんなときには，自分で修理しようとはせず，Bit－INNなどに連絡した方が良いでしょう．テストは，次の順序で行っています．

① Main RAM ·························64KB
② Graphics VRAM (Green) ·········16KB
③ Graphics VRAM (Red) ···········16KB
④ Graphics VRAM (Blue) ···········16KB
⑤ Text RAM ·························32KB

テストにかかる時間は約12分です．無事テストが終了しますと，

Test Complete!

と表示されます．マシンをスタートさせるにはRESETボタンを押します．テストすることによって，システムで使用しているRAMエリア内のデータが壊されてしまうので，コールド・スタートをかけなければならないのです．

## ■Vコマンド（Verify tape＝ベリファイ・テープ）

入力形式　　V　ファイル名

例 1　h］V↵　（最初に見つけたファイルとメモリのデータを比較）
例 2　h］V TEST↵　（TESTというファイルをサーチして，見つけたらメモリのデータと比較します）

Vコマンドは，カセットテープのデータとメモリのデータを比較するコマンドです．カセットテープに無事データをセーブできたかどうかをチェックするときに使用します．

もし，エラーが発見された場合は“？”を表示しますので，もう一度セーブし直す必要があります．

▼画面
①ファイルネームをつけないときの画面
②ファイルネームをつけたときの画面
③エラーが発見されたときの画面

## ■Wコマンド（Write tape＝ライト・テープ）

入力形式　　W　ファイル名，セーブ開始番地，セーブ終了番地

例 1　h］WTEST，D000，D0FF↵（TESTというファイルネームをつけた場合）

例2　h］W, D000, D0FF↵（ファイルネームをつけなかった場合）

Wコマンドは，指定したエリアのメモリのデータをカセットテープにセーブ(書き込み)するコマンドです．ファイルネームはつけなくても可ですが，後のことを考えるとつけた方が便利でしょう．

▼画面
①ファイルネームを指定しなかったときの画面
②ファイルネームをTESTと指定したときの画面

## ■Xコマンド（eXamine＝イグザミン・レジスタ）

入力形式　　X　レジスタ名

例1　h］XA↵　（Aレジスタが表示されます）

例2　h］X↵　（すべてのレジスタが表示されます）

XコマンドはCPUのレジスタを表示したり，変更したりするのに使用します．このコマンドは他のコマンドと組み合わせて，プログラムをデバッグするときに有効に働いてくれます．レジスタのデータを見ることができるので，現在のプログラム・カウンタのデータを読めば，どこのアドレスでストップしているかがわかるからです．

これを利用してGコマンドで，ブレイク・ポイントを設定するとデバッグできます．

ここで，使用されているレジスタの記号を**表1-9**に示します．

**表1-9　Xコマンドで使用されるレジスタの記号**

| | | |
|---|---|---|
| A, A' | アキュムレータ | 8ビット |
| B, B' | BCペアレジスタ | 16ビット |
| D, D' | DEペアレジスタ | 16ビット |
| H, H' | HLペアレジスタ | 16ビット |
| IX | IXインデックス・レジスタ | 16ビット |
| IY | IYインデックス・レジスタ | 16ビット |
| I | Iレジスタ | 8ビット |
| F, F' | フラグレジスタ | 8ビット |
| PC | プログラム・カウンタ | 16ビット |
| SP | スタック・ポインタ | 16ビット |

（注）'が付いているのは補助レジスタを示します．

B，D，Hはそれぞれ16ビット・ペアレジスタとして扱われていますので，8ビット分のレジスタを変更するときは，変更しない残りの8ビット分のデータも入力する必要があります．

レジスタ名を入力しますと，そのレジスタのデータを表示して新しいデータの入力待ちになります（これはSコマンドと同じ動作をします）．

```
h]X
A :00 F :PZ---E-- B :0000 D :EDCC H :0001 A':00 F':PZ---E-- B':0000 D':FF7B
H':0911 IX:2080 IY:7DC9 I :F3 PC:0000 SP:E5F9
```

**▼画面 1**

すべてのレジスタを表示させたときは，2行にわたって出力されます．

```
h]XA
A :00=3E
```

**▼画面 2**

Aレジスタのデータを変更したときは，Aレジスタのもとのデータも表示されます．

```
h]XA
A :00=
F :PZ---E--=
B :0000=
D :EDCC=
H :0001=
A':22=
F':MZ---O--=
B':0000=
D':FF7B=
H':0342=
IX:1206=
IY:0000=
I :F3=
PC:0000=
SP:E5F9=
```

**▼画面 3**

各レジスタのデータを変更しないようにスペース・キーを押していったときは見やすくなります．

## ■ HELP コマンド（HELP＝ヘルプ）

[入][力][形][式]　　HELP　または　CTRL+A

例1　HELP　（⏎キーは押す必要がありません）

例2　CTRL＋A　（⏎キーは押す必要がありません）

HELP コマンドは，モニタを使用しているときにモニタコマンドなどを忘れた場合，使用するものです．[HELP] または [CTRL]+[A] を押すと，CRT に簡単なモニタコマンドの種類とセットするパラメータが表示されます．

このヘルプ機能は，ちょっとモニタコマンドを忘れたり，マニュアルを引くのが面倒なときに使用できますが，ヘルプ機能でモニタコマンドをすべて理解することはできません．

```
Monitor has following commands.
a <address>                    assemble source text lines.
bh or bq                       select radix (hexa decimal or octal).
d <start>,<end>                dump contents of memory.
e <start>                      change contents of memory by screen editor.
f <start>,<end>,<const>        fill memory by the constant.
g <start>,<break1>,<break2>    execute user program.
i <port>                       read input port.
l <start>,<end>                dis-assemble program in memory.
m <start:s>,<end:s>,<start:d>  move contents of memory block(s=src,d=dest).
o <port>,<new value>           output new value to the port.
p                              toggle print switch.
r <file name>                  read cassette file.
s <address>                    substitute memory contents.
tm                             test memory.
v <file name>                  verify cassette file.
w <file name>,<start>,<end>    write cassette file
x or x <register name>         dump all CPU registers or change the register.
CNTL-b                         return to BASIC.
CNTL-d <dr>,<sur>,<tr>,<sec>   dump contents of sectors.
              [,<tr>,<sec>]    <sur> is optional.
CNTL-r <dr>,<sur>,<tr>,<sec>   read sectors into memory.
             ,<start>,<end>
CNTL-w <dr>,<sur>,<tr>,<sec>   write contents of memory into sectors.
             ,<start>,<end>
h]^b
Ok
```

**▼画面**

ヘルプ機能が表示された画面．ただしディスクバージョンでは，CRT表示の初めの部分がスクロールして出てきません．

## ■ CTRL-B コマンド（リターン）

入力形式　　　CTRL＋B

例　h］CTRL＋B

このコマンドを使用すると$N_{88}$-BASICモードに戻りますが，BASICプログラムをこのままの状態で使用するときには注意が必要です．モニタでマシン語を操作するときには，初めに指定したマシン語エリアのみに使用してください．$N_{88}$-BASICのシステムで使用しているメモリ・エリアを勝手に操作すると，

$N_{88}$-BASICモードに戻ったときに動作が保証されなくなってしまいます．最悪の場合は，暴走ということにもなりかねません．

## ■CTRL-Dコマンド（ダンプ・ディスク）

**入力形式**　CTRL+D　ドライブ番号，サーフェス番号（ディスケットの表裏），表示開始のトラック番号，表示開始のセクタ番号，表示終了のトラック番号，表示終了のセクタ番号

（注）トラック，セクタ番号は16進数でなければなりません．

例1　h］CTRL+D1，0，0，1　（ドライブ1，サーフェス0，表示開始のトラック番号0，表示開始のセクタ番号1で，表示開始のセクタのみ表示します）

例2　h］CTRL+D1，0，0，1，0，13　（トラック番号が0でセクタの1〜13まで表示します）

このコマンドは，フロッピィディスクのデータをCRTに表示させるものです．データがディスクにセーブされたかどうか確かめたり，ディスクのどこに書き込まれたのかを調べたりすることができます．さらに，他のディスクの解析にも使用できます．しかし，フロッピィディスクに関しての正しい知識を持っていないと，ディスケットを壊すことにもなりかねません．ただ，このコマンドはデータを見るだけなので，それほど心配する必要はありません．

なお、表示終了トラック番号と表示終了セクタ番号を指定しないときは，表示開始セクタ1セクタのみの表示になります．

```
h]^d1.0.0.1
Track 00.Surface.00.Sector 01
0000 18 01 03 21 00 C1 AF 32 B4 EC 01 02 00 11 11 01      ! ﾁｯ2ｴ●
0010 3A 5D EF B7 20 04 06 02 1E 1B CD F4 C0 30 08 3A     :]\ｷ       ﾍ日ﾀ0 :
0020 B4 EC FE 03 20 F4 C9 3A 61 EF B7 28 10 3A D7 79     ｴ●    日ﾉ:a\ｷ(  :ﾗy
0030 FE 34 38 09 DB 31 E6 80 20 03 CD B9 C1 3A 62 EF      48 ﾛ1       ﾍｹﾁ:b\
0040 B7 CA 00 C1 3E 0B CD C9 37 3E 07 CD D2 37 3E EF     ｷﾊ ﾁ> ﾍﾉ7> ﾍﾒ7>\
0050 CD D2 37 AF CD D2 37 3E 01 CD D2 37 CD 47 38 2F     ﾍﾒ7ｯﾍﾒ7> ﾍﾒ7ﾍG8/
0060 32 03 88 3A 5D EF FE 02 30 26 3A 03 88 B7 CA 00     2 | :]\  0&: | ｷﾊ
0070 C1 FE 10 28 16 3A 5D EF FE 03 20 05 3E 09 32 5D     ﾁ  ( :]\     > 2]
0080 EF 0E 09 CD 80 C1 CD AE C1 18 75 CD A3 C1 18 70     \  ﾍ_ﾁﾍｮﾁ uﾍ｣ﾁ p
0090 3A 02 C0 4F CD 80 C1 3A 02 C0 FE 09 28 D7 F5 3A     : ﾀOﾍ_ﾁ: ﾀ  (ﾗ時:
00A0 03 88 FE 11 20 05 3E FF 32 FD C1 F1 FE 03 28 DB      |     > 2 ﾁ円  (ﾛ
00B0 3A 03 88 B7 28 4A CD 99 C1 18 45 3A 5D EF F5 3E     : | ｷ(Jﾍ┐ﾁ E:]\時>
00C0 01 32 5D EF 37 3E FF CD 34 39 F1 32 5D EF C9 00      2]\7> ﾍ49円2]\ﾉ
00D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00F0 00 00 00 00 AF 3C E5 D5 C5 CD 9A 36 C1 D1 E1 C9          ｯ<ﾕﾅﾍ 6ﾁﾑﾄﾉ
```

**▼画面**

ドライブ1，サーフェス0，トラック0，セクタ1のときの画面で，IPL というプログラムが入っています．

## ■ CTRL-R コマンド（リード・ディスク）

入力形式　CTRL＋R ドライブ番号，サーフェス番号，リード開始のトラック番号，リード開始のセクタ番号，｝ディスクドライブ
書き込み開始アドレス，書き込み終了アドレス→メモリ

例　CTRL＋R1，1，5，1，B000，B0FF
（ドライブ1，サーフェス1，リードトラック5H，リードセクタ1H，リード開始アドレス B000H，リード終了アドレス B0FFH）

このコマンドは，ドライブ番号，サーフェス番号，トラック番号，セクタ番号で指定されたセクタから，書き込み開始アドレス，終了アドレスで指定されたメモリの番地へデータをロードします．1セクタは128バイトですが，これより長いプログラムでも長さに応じてロードしてくれますので，リード開始のセクタのみを指定すればよいのです．128バイトより短くても，メモリのリード開始アドレスと終了アドレスまでのバイト数しかロードされず，他のメモリ・エリアは変更されませんので安心です．

```
h]^r1.1.5.1.D000.D0FF ――――① 
h]DD000.D0FF ――――――――② 
D000 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 00   ABCDEFGHIJKLMNO
D010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

**▼画面**

①ドライブ1，サーフェス1，トラック5，セクタ1からリードして，D000H 番地から D0FF 番地までにロードしたときのものです．

②ロードされたかどうかDコマンドで確めたものです．

## ■CTRL-W コマンド(ライト・ディスク)

**入力形式**　CTRL＋Wドライブ番号，サーフェス番号，セーブ開始トラック番号，セーブ開始セクタ番号，セーブ開始アドレス，セーブ終了アドレス→メモリ（ディスク ドライブ）

例　CTRL＋W1，1，5，1，D000，D0FF
（ドライブ1，サーフェス1，セーブ開始トラック番号5，セーブ開始セクタ番号1，セーブ開始アドレスD000H，セーブ終了アドレスD0FFH）

このコマンドは，フロッピィディスクへメモリのデータをセーブします．カセットテープと原理は同じですが，フロッピィディスク上に細かい番地が割り当てられているために，細かい点まで指定してやらなければならない点がカセットテープと異なります．

1セクタは256バイト単位ですが，指定したメモリのエリアだけセーブしますのでバイトの大きさを気にする必要はありません．256バイトより大きいものは，自動的に数セクタを使用します．

```
h]^w1.1.5.1.D000.D0FF ──①
h]^d1.1.5.1 ──②
Track 05.Surface 01.Sector 01
0000 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 00 ABCDEFGHIJKLMNO
0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

▼画面
①ドライブにセーブしたときのものです．
②CTRL＋Dコマンドを使用して，セーブされたかどうか確めたものです．

モニタコマンドの説明を一通りしてきましたが，中にはあまり使用しないコマンドもありますので，必要によって覚えていけば十分でしょう．

*Section 5*

# アセンブリ言語を使うために

Base

## 人間の言葉に近いアセンブリ言語

マシンが直接に命令を受けつけてくれるのは，0と1の組み合わせの2進数だけです．2進数はマシンには便利ですが，そのままの形で理解することは人間にとって困難です．マシン語を2進数から16進数に変換して理解しようとしたのですが，やはり16進数でも理解は困難でした．

人間が理解しやすい文字列の記号を定めてマシン語表現に使用すれば，数字を扱うよりは数段理解しやすくなります．そこで考え出されたのが「アセンブリ言語」です．アセンブリ言語は，コンピュータが実行できる1命令に対して1つの記号が対応しているように作られており，しかもその記号はマシンの命令の動作を理解できるような文字列を使用しています．

たとえば，

01110110B　2進数表現――――――①

76H　16進数表現――――――②

＊HALT　アセンブリ言語表現――③（＊英語で，止めるという意味）

これらの命令の型はすべて同じ命令で「マシン(CPU)の動きを全てストップせよ」というものです．この3つの表現の中でどれが一番人間に理解しやすいかは一目瞭然ですね．③の“HALT”は何かを停止させることだなと頭に浮ぶことでしょう．しかし，①や②の表現では何のことを言っているのかさっぱりわかりません．アセンブリ言語はだいたいの命令の内容を大まかに知ることができるので覚えるのに都合がいいのです．

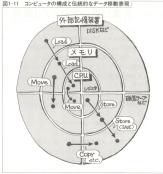
このように人間が理解しやすいようにした文字列の記号のことを「ニーモニック」と呼んでいます．ニーモニックはマシンの命令を表しています．インテル社とザイログ社とでは，同じマシンの命令でもニーモニックが違います．たとえば“HALT”はザイログ形式ですが、インテル形式では“HLT”となっています．

# CPUによるアセンブリ言語の違いと共通点

昔から使用されてきた伝統的な表現を知ることは，いろいろな点でアセンブリ言語習得に役立つものです．この伝統的な表現について説明しましょう．

## ■伝統的なデータ移動の表現

まず，図1-11のような構成のコンピュータを考えます．ここで，CPUを中心として外側から内側に向ってデータが転送されることを“Load”と言います．

図1-11　コンピュータの構成と伝統的なデータ移動表現

逆に，CPUから外側に向ってデータが転送されることを“Store”と言います．レジスタAからレジスタBへの転送のように、同じレベルでの転送は、“Move”と言います．また，外部記憶装置間のデータ転送は“Copy”などの表現を用います．それは，外部記憶装置間では直接データ転送を行えず，コンピュータ本体を使用して中継しなければならないからです．

## ■親しみインテル形式，すっきりザイログ形式

インテル形式のアセンブリ言語は伝統的な表現を数多く反映させていますので，昔からコンピュータに馴れ親しんだ人には覚えやすく，ザイログ形式のア

センブリ言語より使いやすいと思われます．それに対してザイログ形式は，Z-80の命令セットが大きく複雑なために“Move”，“Store”などの表現を捨て，すべて“Load”一言で済ませており，数の少ないすっきりとしたものになっています．といっても，全面的に伝統的な表現を捨て去ったわけではありませんが…．

このようにアセンブリ言語は各CPUによって違いがありますが，共通の部分も多いのです．この勘どころを押えると，アセンブリ言語を早く習得できると思います．

## アセンブリ言語表現の約束事

キーボードにある文字であれば，どんな文字でもアセンブリ言語表現に使えそうですが，アセンブラ（後で説明します）を使用したりするために，使える文字は限られています．しかも，記号としての文字列とデータとしての文字列とに分けて考えておく必要があります．一般的に使用できる文字は次のようなものです．これ以外の文字を使うことはできませんので，しっかりと覚えておいてください．

### ■使用できる文字

① A～Zの大文字（場合によっては小文字も使用できる）

② 0～9の数字

③ #，a，?，-，などの記号も使えるものもある

こう見ると，アセンブリ言語で使用できる文字は，BASICなどに比べて非常に限定されていることがわかります．わかりやすくするために使用する文字の種類を少なくしているのだと思われます（もちろん，他にもいろいろ理由はありますが……）．

プログラムを読みやすくするためには，記号としての文字列の構成に注意することが大切になってきます．文字列を作るとき，英語の綴りにしたりローマ字の綴りにしたりしますが，コメントを含めてどちらかに統一した方が良いでしょう．特にコメントは小文字が使用できたらその方が便利になります．文字列とデータ列の約束事は一般的に次のようになっています．

### ■文字列とデータ列との約束事

●文字列…………①初めの文字は必ずアルファベット

②6文字以内

| 正しい例 | 誤った例 |
|---|---|
| ABCDEF | 1ABCDE |
| LOOP | ABC DE |
| LOOP3 | ␣ABC3 |

●**数値データ列**…①初めの文字は0～9までの数字．もしA～Fが使用されたときは0を頭につける
②2進数，8進数，16進数の区別をはっきりつける

| 正しい例 | 誤った例 |
|---|---|
| 13B5H | 1D4C |
| 1010B | 1410B |
| 0F300H | F300H |

### ■ 0がついたら数値データ

文字列の第1文字はアルファベットから始まっていなければなりません．これは，文字列と数値データ列とをはっきり区別するためです．これは数値データ列にも言えることで，16進数を扱う場合にA～Fまでのアルファベットを使用していますので，数値データの第1文字にA～Fがくることもあります．そこで，文字列と区別するために，第1文字がA～Fになったときには，その前に0をつけて数値データであることを表示します．

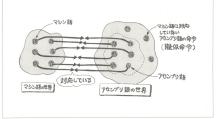
また，文字列などは予約記号（予約語）と呼ばれるものもあります．予約記号の中にはCPUのレジスタ名も含まれますので，“BC”，“HL”などの文字列は使ってはなりません．スペースも記号と記号の区切りとして使用していますので，使用することはできません．

## マシンにない擬似命令

アセンブリ言語の命令はマシンの命令と1対1で対応しているわけですから，当然，アセンブリ言語の命令はマシンの命令数を含んでいるのですが，その他にマシンにはない命令をいくつか持っています．

マシンにはない命令を擬似命令と言います．擬似命令はアセンブリ言語で書いたプログラムをわかりやすくするのと，アセンブラ（後で説明します）に連絡するために用いられます．

アセンブリ言語の世界は**図1-12**のようになっています．

図1-12　アセンブリ言語の世界

## ■基本的な擬似命令

基本な擬似命令には，**表1-10**のようなものがあります．

表1-10　基本的な擬似命令

| 擬似命令 | 由来語 | 意味 |
|---|---|---|
| ORG | Origin | マシン語のスタート・アドレス (本来はロケーションカウンタ) |
| END | End | アセンブリ語の終了 |
| DB, DEFB | Define Byte | 指定された番地に1バイト・データをセット |
| DW, DEFW | Define Word | 指定された番地に2バイト・データをセット |
| DC | Define Character | DBと同じ |
| DC, CONST | Define Constant | 指定した番地に定数をセット |
| DS | Define Storage | 指定された番地から指定されたバイト数を確保 |
| EQU | Equate | 文字列に数値データを与える |
| DATA | Data | DBと同じ |

ここに出てきた擬似命令を知っていないと，実用的なプログラムをアセンブ

リ言語で書けませんので必ず覚えておきましょう.

## アセンブリ言語のかたち

アセンブリ言語のプログラムの書き方は昔からあまり変っておらず，全く共通しているといえるくらい守られています.

命令は1行（1ライン）にひとつだけ書き，原則としてマシンの命令に対応するように書いていきます．この1行に書かれた命令を文と呼んでいます.

BASICのようにマルチ・ステートメントはできません．次に文の例を示しますので見てください.

a　　　b　　　c　　　d

ラベル：ニーモニック␣オペランド；コメント

### ■ラベルは名札

ラベルは命令などにつけられる名札で，BASICの行番号に相当するものですが，必要のないところにはつけません．ジャンプ命令やコール命令などのときに参照する標識となるものです.

### ■ニーモニック，オペランド

実行命令で，マシンの命令を2つに分けて表しています．ニーモニックはしいて言えば大まかな命令で，オペランドは具体的な命令の操作を表しています.

### ■コメント

コメントは全くの覚え書き用でアセンブラは無視しますので，プログラマが自由に利用できます.

では，具体的な例を示します.

```
ラベル  ニーモニック  オペランド   コメント
START：EQU   0A000H   ；Start Address （STARTという変数に，A000Hをセットする）
       ORG   START                    （マシン語のプログラム開始番地はA000Hとなる）
       LD    B, 03H   ；Move B→03     （Bレジスタに03Hをロードせよ）
LOOP ：LD    A, 41H   ；Move A→41     （Aレジスタに41Hをロードせよ）
       DEC   B        ；Decrement B-1 （Bレジスタから1を引く）
       JP    NZ, LOOP ；Jump Loop     （ループというラベルのあるところへゼロフラグが立ってなければジャンプせよ）
       END
```

このようになりますが，ニーモニックの“LD”はデータを移動せよという命令だけで，具体的にどうせよとは表現していません．どこに，どのデータを移動せよと表現しているのがオペランドです．しかも，オペランドは「,」を中心にして右から左へ移動するように書き表すことになっていますから，すっきりしています．

## ハンド・アセンブルでマシン語マスター

アセンブリ言語で書いたプログラムで，すぐにマシンを動かすことはできません．つまり，マシンはそのままの形ではアセンブリ言語を理解できないのです．マシンは2進数しか受けつけてくれませんから，アセンブリ言語で書いたプログラムを何らかの方法で2進数に変換してやらなければなりません．このアセンブリ言語を2進数に変換することをアセンブル作業と言います．アセンブル作業には，2つの方法があります．

a．手作業で行う方法（ハンド・アセンブル）

b．セルフ・アセンブラ(アセンブラとも言う)プログラムを使用する方法

これを図で表すと**図1-13**になります．

**図1-13　アセンブルの手順**



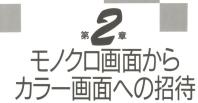
ハンド・アセンブルとは，人間がマシン語の対応表を見ながら，1命令ずつアセンブリ言語からマシン語に変換していく作業のことです．たいへん面倒な作業なのですが，マシン語を習い始めた人は必ずこの作業をやらなければなり

ません．この作業に慣れるに従って，アセンブリ言語に対応するマシン語を覚えていきますから，非常に好都合なものです．

ハンド・アセンブルをするのには，ちょっとした道具が必要です．それは，アセンブリ言語とマシン語との対応表です．この本の付録として対応表をつけましたので，ハンド・アセンブルの際に使ってください．

ハンド・アセンブルは，短いものなら短時間で確実にマシン語に変換できますが，プログラムが長くなるにしたがって，間違えずにマシン語に変換するのが難しくなってきます．その点，アセンブラを使えば，正しくアセンブリ言語で書かれていれば，間違いなくマシン語に変換してくれます．このプログラムを使用すると，面倒でバグを作りやすい作業をしなくて済みます．

この章では，マシン語をマスターするための基礎的な知識について説明しました（フローチャートについては説明していませんが，理解されているものとして省いてあります）．ここまでのことがある程度わかっていると，次章からのことがよく理解できると思います．

# 第2章 モノクロ画面から カラー画面への招待

キャラクタ単位で色をつける

さて，マシン語についてひととおりの知識を獲得したところで，この章からは本書のメインである画面操作の説明に入っていきます．

画面に自分の好きな色をつけられたら楽しいですね．まず，どうやって画面に色を出しているのかを知らないと話にならないので，そこから説明していきましょう．アトリビュートエリアを利用して，テキスト画面上の文字を反転させたり点滅させたりします．そして，キャラクタ単位で色をつけたり，バックグランドに色をつけたりといったテキスト画面のカラー化をマスターしましょう．

*Section 1*

# CRTのカラーの原理

## CRTの基本は3原色

家庭にあるTVはほとんどカラーTVです．コンピュータ用のTV(CRTとも言います)も家庭用TVを利用したものなので原理は同じです．TVがカラーになるのは光の3原色を利用したもので，青，赤，緑の組み合わせによってです．TVのスクリーンの近くには青，赤，緑の螢光塗料が塗ってあり，この3原色の光り方で色がつくことになります．**図2-1**のようになりますが，これでは黒を含めて8色しか出ません．しかし、家庭用のTVは無限色に近い色を出すことができます．家庭用TVでは，青，赤，緑の強弱を変えることによって様々な色を作り出すのです．

### ■かけあわせれば色いろいろ

カラーCRTでは，青，赤，緑の強弱を変えることができません．それで色は8色のみになってしまいます．しかし，工夫次第では中間色も可能です．後章のグラフィックス画面の章で説明しましょう．

**図2-1**からわかるように，緑と青を発光させれば水色になり，緑と赤を発光させると黄色になります．また，青，緑，赤を発光させると白になります．キャラクタ単位で色をつけるとき，この原理を知っていないと，希望の色を出すことができません．

図2-1　3原色による色

*Section 2*

# テキスト画面のVRAMを知る

## ディスプレイエリア(表示用)とアトリビュートエリア(装飾用)

CRT画面に文字を表示させるためには，VRAMを操作する必要があります．このVRAMは，CRTの画面にメモリの番地を割り当てたものです．したがって，VRAMにデータを書き込んだり読み出したりすることが自由にできます．

VRAMの番地とCRTの画面上の位置関係は**図2-2**のようになっています．このようにVRAMは画面表示用（ここではディスプレイエリアと呼ぶことにします）と表示には直接タッチしない画面装飾用（アトリビュートエリア）とに分けられます．ディスプレイエリアはCRTの画面と一致しており，この番地にキャラクタ・コードを書き込めば画面上に表示されます．アトリビュートエリアは，表示されている文字をリバース（反転）したり、ブリンク（点滅）したり，あるいはシークレット（表示させない）したり，アンダラインとアッパラインを引いたりすることに使用するものです．また，カラーモードのときは文字に色をつけるためにも使用されます．

アトリビュートエリアの使い方はプログラムを作りながら説明していきますが，ディスプレイエリアの番地は付録を利用してください．

図2-2　VRAMの番地とCRT画面上の位置関係

| 画面表示用（CRTの画面の位置と一致している）ディスプレイエリア |  |  | 画面装飾用（表示されない部分で，表示されているキャラクタに変化をつける時に使用される部分）アトリビュートエリア |  |  |
| --- | --- | --- | --- | --- | --- |
| F3C8 | ～ | F417 | F418 | ～ | F43F |
| F440 | ～ | F48F | F490 | ～ | F4B7 |
| ～ | ～ | ～ | ～ | ～ | ～ |
| FE90 | ～ | FEDF | FEE0 | ～ | FF07 |
| FF08 | ～ | FF57 | FF58 | ～ | FF7F |
| 80バイト |  |  | 40バイト |  |  |

*Section 3*

# 画面表示に必要なキャラクタ・コード

Text VRAM

## カナやアルファベットも数値で扱う

コンピュータはアルファベットやカナ文字などもすべて数値として扱っています．したがって文字などは対応する数値を用いなければ，希望の文字が表示されません．その決められた数値をキャラクタ・コードと言います．たとえばAという文字を表示させるには41Hを使用しなければなりません．パソコンで使用されているキャラクタ・コードはASCII（アスキー：米国の規格）コードに準拠しています．準拠しているといっても，同じコードなのは半分以下です．なお，大型コンピュータではASCIIコードは使用していませんので注意してください．

PC-8801mkIISR（PC-8801，PC-8001）のキャラクタ・コードは**表2-1**のようになっています．

**表2-1　キャラクタ・コード**

| 上位<br>下位 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | $D_E$ | | 0 | @ | P | | p | ▁ | ┴ | | ー | タ | ミ | ═ | ╳ |
| 1 | $S_H$ | $D_1$ | ! | 1 | A | Q | a | q | ▂ | ┬ | 。 | ア | チ | ム | ╞ | 円 |
| 2 | $S_X$ | $D_2$ | " | 2 | B | R | b | r | ▃ | ┤ | 「 | イ | ツ | メ | ╪ | 年 |
| 3 | $E_X$ | $D_3$ | # | 3 | C | S | c | s | ▄ | ├ | 」 | ウ | テ | モ | ╡ | 月 |
| 4 | $E_T$ | $D_4$ | \$ | 4 | D | T | d | t | ▅ | ▔ | 、 | エ | ト | ヤ | ◢ | 日 |
| 5 | EQ | $N_K$ | % | 5 | E | U | e | u | ▆ | ▀ | ・ | オ | ナ | ユ | ◣ | 時 |
| 6 | $A_K$ | $S_N$ | & | 6 | F | V | f | v | ▇ | ▌ | ヲ | カ | ニ | ヨ | ◥ | 分 |
| 7 | $B_L$ | $E_B$ | ' | 7 | G | W | g | w | █ | ▐ | ァ | キ | ヌ | ラ | ◤ | 秒 |
| 8 | $B_S$ | $C_N$ | ( | 8 | H | X | h | x | ▏ | ┌ | ィ | ク | ネ | リ | ♠ | |
| 9 | $H_T$ | $E_M$ | ) | 9 | I | Y | i | y | ▎ | ┐ | ゥ | ケ | ノ | ル | ♥ | |
| A | $L_F$ | $S_B$ | * | : | J | Z | j | z | ▍ | └ | ェ | コ | ハ | レ | ◆ | |
| B | $H_M$ | $E_C$ | + | ; | K | [ | k | { | ▌ | ┘ | ォ | サ | ヒ | ロ | ♣ | |
| C | $C_L$ | → | , | < | L | ¥ | l | ¦ | ▋ | ╭ | ャ | シ | フ | ワ | ● | |
| D | $C_R$ | ← | − | = | M | ] | m | } | ▊ | ╮ | ュ | ス | ヘ | ン | ○ | |
| E | $S_O$ | ↑ | . | > | N | ^ | n | ~ | ▉ | ╰ | ョ | セ | ホ | ゛ | ╱ | |
| F | $S_i$ | ↓ | / | ? | O | _ | o | | ┼ | ╯ | ッ | ソ | マ | ゜ | ╲ | |

Section 4

# アトリビュートエリアの属性を知る

## 属性を変化させる

ディスプレイエリアにキャラクタを表示させるには，ディスプレイエリアの番地にキャラクタ・コードを書き込みます．しかし，キャラクタを反転（リバース）させたり点滅（ブリンク）させたりすることはできません．そこでキャラクタ・コードとは直接関係ない情報をアトリビュートエリアに書き込んで反転や点滅をさせます．

### ■属性変更は20回まで

アトリビュートエリアは1行につき40バイトで構成されており，1回の属性を変化させるたびに2バイト使用されます．したがって，属性は1行について20回までしか変えることができません．

アトリビュートエリアの構成を1行目を例にとって考えてみましょう．**図2-3**のようになります．どの行も番地が違う以外すべて同じです．

初めの1バイトで何行目から属性を変化させるかを指定し，次の1バイトでどのように変化させるかを指定します．もし、1行全部を反転させるのであれば，最初の2バイトだけ指定すればその行全部が反転します．

**図2-3　1行目のアトリビュートエリア**

| F418H | F419H | F41AH | F41BH | | F43EH | F43FH |
|---|---|---|---|---|---|---|
| 桁数 | 属性 | 桁数 | 属性 | ………………………………… | 桁数 | 属性 |
| 1バイト | 1バイト | | | | | |
| 何桁目から属性を変化させるか指定する | ここで反転や点滅などの情報を与える | | | | | |

## ■属性の指定方法

次に属性の指定方法について考えてみましょう．属性のデータはモノクロモードとカラーモードでは違ってきます．カラーモードの場合は，反転や点滅などのほかにカラー指定しなければなりませんので，モノクロモードの属性指定よりは複雑になります．属性コードの構成は**図2-4**のようになります．

**図2-4　属性コードの構成**

①モノクロモード

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 ビット |
|---|---|---|---|---|---|---|---|
| GRAPHIC<br>CHAR | 常に0 | アンダライン | アッパライン | 常に0 | リバース | ブリンク | シークレット |
| 0=キャラクタ・モード<br>1=グラフィック・モード | | 0=表示しない<br>1=表示する | 0=表示しない<br>1=表示する | | 0=変化なし<br>1=リバースする（反転） | 0=変化なし<br>1=ブリンクする（点滅） | 0=表示する<br>1=シークレットする（表示しない） |

②カラーモード　その1

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 ビット |
|---|---|---|---|---|---|---|---|
| 常に0 | 常に0 | アンダライン | アッパライン | 0 | リバース | ブリンク | シークレット |
| | | 0=表示しない<br>1=表示する | 0=表示しない<br>1=表示する | ※ | 0=変化なし<br>1=リバースする | 0=変化なし<br>1=ブリンクする | 0=表示する<br>1=シークレットする |

③カラーモード　その2

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 ビット |
|---|---|---|---|---|---|---|---|
| 緑 色 | 赤 色 | 青 色 | GRAPHIC<br>CHAR | 1 | 常に0 | 常に0 | 常に0 |
| 0=OFF<br>1=ON | 0=OFF<br>1=ON | 0=OFF<br>1=ON | 0=キャラクタ・モード<br>1=グラフィック・モード | ※ | | | |

（ビット7〜5）カラーコードをこの3ビットで指定する

※カラーモードにおいては，3ビット目が0か1かによって属性が変化する．

たとえばモノクロモードで1行を反転させたいときは，属性の番地（たとえば1行目ならF419H）に04H（00000100B）を書き込めばよいことになります．この属性の扱い方は次の項で練習してみましょう．

## 使い方あれこれ

この属性の利用のし方はたくさんあります．たとえば，リバースはメニュープログラムを選択するときに，選択したプログラムを反転させるのに使用できます．ブリンクはオペレータに注意を促すときに使用できます．またシークレットは，キーワードやパスワードのように機密性を必要とするものを表示させないために使うことができます．他にいろいろなプログラムで，アトリビュートエリアの属性は使用できるので利用してください．

# Section 5

# モノクロモードでアトリビュートエリアを操作する

## 文字"A"の属性を変化させる

図2-5 画面表示

アトリビュートエリアの属性がわかったところで,その属性を使う練習をしてみましょう.最初の行に「A」という文字を表示させてから,リバース,ブリンク,シークレットに属性を変えて,最後にこの3つの属性を混在させてみようというわけです.練習の順番は次のようにしてみましょう.

練習1　リバースのプログラム
練習2　ブリンクのプログラム
練習3　シークレットのプログラム
練習4　3つの混在するプログラム

## 文字をリバース(反転)させる

まず,画面の設計から考えていきましょう.手順は,F3C8H番地から「A」を80個表示した後に,アトリビュートエリアのF418H番地に00Hをセットし,F419H番地にリバースさせるデータ04Hをセットし,モニタへ戻ればいいことになります(図2-6, P.78).

### ■使用するアドレスとデータ

では,ここで使用するアドレスやデータをまとめておきましょう.

| | |
|---|---|
| ○プログラムのスタート・アドレス | B900H |
| ○CRTのスタート・アドレス | F3C8H |
| ○アトリビュートエリアの桁数アドレス | F418H |
| ○アトリビュートエリアの属性アドレス | F419H |

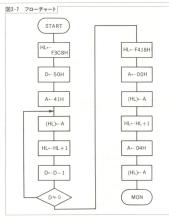



図2-6　リバース

図2-7　フローチャート

○モニタのアドレス　E826H
○属性データ　04H
○書き込みキャラクタ「A」　41H

ここで属性データが04Hになっているのは，3ビット目が1になればリバースすることになるので，他はすべて0になるようにするためです．属性コードをもう一度見てください．

## ■フローチャート＆プログラム

次にフローチャートを作成します．フローチャートを**図2-7**に示します．

プログラムリストは**リスト2-1**のようになります．プログラムの説明をしましょう．まず，HLペアレジスタにCRTのスタート・アドレスをセットし，Dレジスタに80回ループを繰り返す50Hをセットします．Aレジスタに表示するキャラクタ・コードの41H「A」をセットしたら，HLペアレジスタが指定するVRAMのディスプレイエリアにロードし、それを50H繰り返させます．

ループが終ったならばアトリビュートエリアのアドレスをHLペアレジスタにセットし直します．このプログラムではセットしなくてもHLペアレジスタのデータはF418H（1行目のアトリビュートエリアの番地）になっています

**●リスト2-1　Aをリバースさせる**

```
                 ;********* Monochrome Text Mode ***********
                 ;
                 ; レンシュウ ノ 1
B900             START: EQU   0B900H          ;START ADDRESS
E826             MON:   EQU   0E826H          ;MONITOR ADDRESS
F3C8             CRTAD: EQU   0F3C8H          ;V-RAM START ADDRESS
F418             SKETA: EQU   0F418H          ;V-RAM  AV ADDRESS
0004             AV:    EQU   04H             ;REVERSE DATA 04H
0041             CHAR:  EQU   41H             ;CHAR DATA 'A'
                 ;
                        ORG   START
                 ;
B900 21C8F3             LD    HL,CRTAD
B903 1650               LD    D,50H           ;1 LINE DATA 80KETA
B905 3E41               LD    A,CHAR          ;DISPLAY CHAR CODE
B907 77          LOOP1: LD    (HL),A
B908 23                 INC   HL
B909 15                 DEC   D
B90A C207B9             JP    NZ,LOOP1
B90D 2118F4             LD    HL,SKETA
B910 3E00               LD    A,00H           ;DISPLAY NO START KETA
B912 77                 LD    (HL),A
B913 23                 INC   HL
B914 3E04               LD    A,AV            ;V-RAM AV DATA
B916 77                 LD    (HL),A
B917 C326E8             JP    MON             ;MONITOR
                 ;
B91A                    END
```

が，いつもループが終ったときにアトリビュートエリアのアドレスになっているとは限りませんので，セットし直した方がベターです．それからF418H番地に0桁目から表示するデータの00Hをセットし，F419H番地にリバースするための属性データをセットし，モニタに戻ります．

### ■実行するには

実行するときは，EコマンドやSコマンドを使用してB900H番地から入力して実行させてください．写真のように1行にAが表示されて反転している画面になるはずです．

### ■プログラムをディスクにセーブするには

なお，このプログラムをディスクにセーブするには，BASICモードに戻ってBSAVEコマンドを使用します．

BSAVE　"mono1. B", &HB900, &H19H ⏎

で，**プログラム2-1**がセーブされます．「DUAD-88」を使用している場合はホット・スタート（STOP＋リセット・ボタン）してBSAVEしてください．

## リバースに使用したマシン語命令

### ■LD（ロード）命令

一番多く使用される命令で，8ビットや16ビットのデータをCPUのレジスタ同士やメモリの間で移動させる命令です．データの移動といっても移動先のデータは変化しませんから，データのコピーと覚えた方が良いでしょう．

LD r, n

（注：rは汎用レジスタ名を表し，nは8ビット・データを表します）

書式　　LD　r, n

機能　8ビット・データをrで指定されるレジスタにロードします．
例　LD A, 41H
Aレジスタに41Hをロードします．

### LD (HL), r

書式　LD (HL), r
機能　rレジスタのデータをHLペアレジスタで示されるメモリの番地にロードします．
例　LD (HL), A
HLペアレジスタの内容をF3C8Hとすると

Aレジスタのデータ(41H)がF3C8H番地へロードされます．

### LD pr, nn　(注：prはペアレジスタHL, DE, BCを表します)

書式　LD pr, nn
機能　指定するペアレジスタに2バイトのデータをロードします．
例　LD HL, 0F418H
Hレジスタへ F4H (上位バイト) を，Lレジスタに18H (下位バイト) をロードします．

## ■ INC(インクリメント)命令とDEC(デクリメント)命令

INC, DEC命令は汎用レジスタに1をプラスしたりマイナスしたりするもので，回数を数えたりするのに多く使用されます．また，DEC命令は，次に説明するJP命令をセットして使用することが多くなります．この命令はペアレジスタでも使用できます．

INC　pr

書式　INC　pr

機能　指定されたペアレジスタに1を加算し，その結果をペアレジスタに入れます（ペアレジスタ ← ペアレジスタ+1）．

例　INC　HL

実行前の HL ペアレジスタの内容　F3C8

実行後の HL ペアレジスタの内容　F3C9

実行の結果，HL ペアレジスタに1が加算されました．

DEC　r

書式　DEC　r

機能　指定されたレジスタから1を減算し，結果をレジスタに入れます．その際，フラグも変化します（レジスタ ← レジスタ+1）．

例　DEC　D

Dレジスタから1を減算します．その結果，たとえばDレジスタの値が00Hになったときは Z（ゼロ）フラグは1になります．

## ■ JP（ジャンプ）命令

JP 命令はプログラムの流れを変えるために使用されます．フラグの内容によって流れも変化するものもあります．

JP　nn

書式　JP　ジャンプ先の番地またはラベル

機能　nn で指定された番地に無条件でジャンプ（プログラム・カウンタに nn 値をロード）します．フラグの影響は受けません．

例　JP　0A000 H

メモリの A000H 番地へジャンプします．

## JP NZ, nn

書式　JP NZ, ジャンプする番地

機能　ゼロフラグが0のときに指定する番地にジャンプします．ゼロフラグが1のとき（レジスタの値が0）に次の命令を実行します．

例
```
DEC  B
JP   NZ, 9000H
DEC  A
```
ゼロフラグが0のときは9000H番地へジャンプし，ゼロフラグが1のときは“DEC　A”の命令を実行します．

この「JP　Z, nn」命令は使用していませんが，「JP　NZ, nn」とは反対の命令ですから，対にして覚えましょう．

## JP Z, nn

書式　JP Z, ジャンプする番地

機能　ゼロフラグが0のときに次の命令を実行します．ゼロフラグが1のとき(レジスタの値が0)は指定する番地にジャンプします．

例
```
DEC  B
JP   Z, 9000H
DEC  A
```
ゼロフラグが0のときは“DEC　A”の命令を実行し，ゼロフラグが1のときは9000H番地へジャンプします．

## プログラムリストの見方について

**リスト2-1**のようなプログラムがこれから先たくさん出てきますので，このプログラムリストの見方を説明しておきましょう．プログラムリストに説明を加えると，**図2-8**のようになります．

### ■EQU命令を使うわけ

EQU命令が多く使用されていますが，次のような理由によるものです．プログラム中の次の箇所に注目してください．

```
CHAR：EQU　41H
　　　　：
　　　LD　A，CHAR（41H）——①
```

①のLD命令では，Aレジスタにキャラクタの“A”で41Hを直接に入力しないで，EQU命令を使って“CHAR”に41Hを入れてから，“CHAR”というラベルでAレジスタにロードしています．一見複雑なようですが，この方が便利なのです．

というのは，今回のプログラムでは1回しか“A”レジスタにキャラクタ・コードをロードしていませんが，2箇所以上でてくる場合は，キャラクタが変わるたびにプログラムを書き直さなければなりません．2箇所ある場合は，2箇所を別のキャラクタ・コードにしなければなりません．

しかし，ラベルで定数（この場合はキャラクタ・コード）を宣言しておき，LD命令などを用いてラベル名でキャラクタ・コードをロードできるように記述しておけば，何箇所あってもEQU命令の定数を一箇所書き直すだけで良いのです．後のことはアセンブラがすべてラベルを定数に置き換えてくれます．

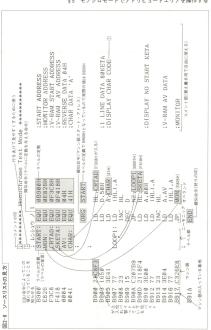
このようにEQU命令を使用することによって，時間の短縮とミス防止をすることができます．つまり，本当のマシン語（擬似命令でない命令）の部分をいじらないので，プログラムがおかしくなることもないのです．

### ■まずはハンド・アセンブルから

アセンブラを使用すると，ハンド・アセンブルでは考えられないほどスピードが上がりますが，初めての人は必ずハンド・アセンブルを行ってください．その方が，遠回りのようでも近道となります．

## 1行ブリンク（点滅）させるには

ブリンクさせるといっても特別なことをしているわけではなく，アトリビュ

図2-8 ソースリストの見方

ートエリアの属性を変えているだけです．**リスト2-2**を見てください．**リスト2-1**のプログラムと違うのは，属性データだけです．つまり，属性データ以外は**リスト2-1**のプログラムと全く同じで良いということです．B900H番地から実行させると，1行ブリンク（点滅）するはずです．

**●リスト2-2　Aをブリンクさせる**

```
                ;********* Monochrome Text Mode ***********
                ;
                ; レンシュウ ノ 2
B900            START:  EQU   0B900H          ;START ADDRESS
E826            MON:    EQU   0E826H          ;MONITOR ADDRESS
F3C8            CRTAD:  EQU   0F3C8H          ;V-RAM START ADDRESS
F418            SKETA:  EQU   0F418H          ;V-RAM  AV ADDRESS
0002            AV:     EQU   02H             ; BLINK DATA 02H
0041            CHAR:   EQU   41H             ;CHAR DATA 'A'
                ;                  ここだけがリスト2-1と違う
                        ORG   START
                ;
B900 21C8F3             LD    HL,CRTAD
B903 1650               LD    D,50H           ;1 LINE DATA 80KETA
B905 3E41               LD    A,CHAR          ;DISPLAY CHAR CODE
B907 77         LOOP1:  LD    (HL),A
B908 23                 INC   HL
B909 15                 DEC   D
B90A C207B9             JP    NZ,LOOP1
B90D 2118F4             LD    HL,SKETA
B910 3E00               LD    A,00H           ;DISPLAY NO START KETA
B912 77                 LD    (HL),A
B913 23                 INC   HL
B914 3E02               LD    A,AV            ;V-RAM AV DATA
B916 77                 LD    (HL),A
B917 C326E8             JP    MON             ;MONITOR
                ;
B91A                    END
```

# 1行を消してしまう〜シークレット

このプログラムも**リスト2-2**と同様，注の部分の属性データが変わっているだけです．リストを示しましょう（**リスト2-3**）．

実行させると1行が消えます．しかし，ディスプレイエリアのF3C8HからF417H番地までは，“A”のキャラクタ・コードである41Hが間違いなく書き込まれているはずです．モニタのEコマンドで確認してみてください．

## ■落し穴や異次元の入口にも使える

ディスプレイエリアにデータが残っていると，使い方によっては面白いことができます．たとえば，パスワードの確認のために使用できますし，ゲームなどでは落し穴や異次元の入口のように，普段は消しておいて近くに来たり触れたりしたときに現れるようにすれば，変化のある画面をつくることができるで

しょう.

●リスト2-3　1行をシークレットする

```
                ;********* Monochrome Text Mode ***********
                ;
                ; レンシュウ ノ 3
B900            START:  EQU   0B900H          ;START ADDRESS
E826            MON:    EQU   0E826H          ;MONITOR ADDRESS
F3C8            CRTAD:  EQU   0F3C8H          ;V-RAM START ADDRESS
F418            SKETA:  EQU   0F418H          ;V-RAM  AV ADDRESS
0001            AV:     EQU   01H             ; SECRET DATA 01H
0041            CHAR:   EQU   41H             ;CHAR DATA 'A'
                ;                   ここだけが違う
                        ORG   START
                ;
B900 21C8F3             LD    HL,CRTAD
B903 1650               LD    D,50H           ;1 LINE DATA 80KETA
B905 3E41               LD    A,CHAR          ;DISPLAY CHAR CODE
B907 77         LOOP1:  LD    (HL),A
B908 23                 INC   HL
B909 15                 DEC   D
B90A C207B9             JP    NZ,LOOP1
B90D 2118F4             LD    HL,SKETA
B910 3E00               LD    A,00H           ;DISPLAY NO START KETA
B912 77                 LD    (HL),A
B913 23                 INC   HL
B914 3E01               LD    A,AV            ;V-RAM AV DATA
B916 77                 LD    (HL),A
B917 C326E8             JP    MON             ;MONITOR
                ;
B91A                    END
```

# 1行にいろいろな属性をもたせる

ここでは，5種類ある属性をすべて使用してみましょう.

## ■反転文字を点滅―リバース＋ブリンク

属性は組み合わせて使用することができますので，リバースとブリンクを組み合わせてみましょう．リバースとブリンクを組み合わせたプログラムを**リスト2-4**に示しますが，このプログラムも**リスト2-1**とほとんど同じです.

●リスト2-4　リバース＋ブリンク

```
                ;********* Monochrome Text Mode ***********
                ;
                ; レンシュウ ノ 4-1
B900            START:  EQU   0B900H      ;START ADDRESS
E826            MON:    EQU   0E826H      ;MONITOR ADDRESS
F3C8            CRTAD:  EQU   0F3C8H      ;V-RAM START ADDRESS
F418            SKETA:  EQU   0F418H      ;V-RAM  AV ADDRESS
0006            AV:     EQU   06H         ;REVERSE AND BLINK DATA 06H
0041            CHAR:   EQU   41H         ;CHAR DATA 'A'
                ;                   ここだけが違う
                        ORG   START
```

```
                    ;
B900 21C8F3               LD    HL,CRTAD
B903 1650                 LD    D,50H         ;1 LINE DATA 80KETA
B905 3E41                 LD    A,CHAR        ;DISPLAY CHAR CODE
B907 77           LOOP1:  LD    (HL),A
B908 23                   INC   HL
B909 15                   DEC   D
B90A C207B9               JP    NZ,LOOP1
B90D 2118F4               LD    HL,SKETA
B910 3E00                 LD    A,00H         ;DISPLAY NO START KETA
B912 77                   LD    (HL),A
B913 23                   INC   HL
B914 3E06                 LD    A,AV          ;V-RAM AV DATA
B916 77                   LD    (HL),A
B917 C326E8               JP    MON           ;MONITOR
                    ;
B91A                      END
```

この属性データは3ビット目（リバースのビット）と2ビット目（ブリンクのビット）を1にするだけで，他のビットはすべて0にします．実行させると反転文字になって点滅を繰り返します．

## ■文字の上下に線をひく―アッパライン＋アンダライン

BASICなどではサポートされていないアッパラインとアンダラインを表示させてみましょう．**リスト2-5**がそのプログラムです．

前のプログラムとの違いは，属性データを30Hにするだけです．実行させると，文字の上と下に線が引かれます．

**●リスト2-5　アッパライン＋アンダライン**

```
                  ;********* Monochrome Text Mode ***********
                  ;
                  ; レンシュウ ノ 4-2 (UPPER & UNDER LINE)
B900              START:  EQU   0B900H    ;START ADDRESS
E826              MON:    EQU   0E826H    ;MONITOR ADDRESS
F3C8              CRTAD:  EQU   0F3C8H    ;V-RAM START ADDRESS
F418              SKETA:  EQU   0F418H    ;V-RAM  AV ADDRESS
0030              AV:     EQU   30H       ; UPPER AND UNDER LINE DATA 06H
0041              CHAR:   EQU   41H       ;CHAR DATA 'A'
                  ;                        ここだけが違う
                          ORG   START
                  ;
B900 21C8F3               LD    HL,CRTAD
B903 1650                 LD    D,50H     ;1 LINE DATA 80KETA
B905 3E41                 LD    A,CHAR    ;DISPLAY CHAR CODE
B907 77           LOOP1:  LD    (HL),A
B908 23                   INC   HL
B909 15                   DEC   D
B90A C207B9               JP    NZ,LOOP1
```

```
B90D 2118F4          LD    HL,SKETA
B910 3E00            LD    A,00H      ;DISPLAY NO START KETA
B912 77              LD    (HL),A
B913 23              INC   HL
B914 3E30            LD    A,AV       ;V-RAM AV DATA
B916 77              LD    (HL),A
B917 C326E8          JP    MON        ;MONITOR
                ;
B91A                 END
```

## ■8種類の属性を1行に表示

属性データを変えて8種類の違ったものを表示させましょう．8種類の組み合わせにしたのは，シークレットと他のものを組み合わせてもラインが表示されているかの確認がとれませんので意味がないからです．意味があるものだけを選ぶと8種類になるわけです．

属性の種類が8種類ありますので，1つの属性に対して10文字分割り当てることができます．画面に**図2-9**のように表示させることにしましょう．

**図2-9　画面構成**

これに対応する属性コードは**表2-2**(P.90)のようになります．

アトリビュートエリアは2バイトで1回属性を変化させることができます．今回は8回属性を変えるわけですから，16バイト使用することになります．初めのバイトに表示する桁の位置を書き込んで，次のバイトに属性データを書き込んでやることを8回繰り返せば良いことになります．プログラムでは，1バイト書き込むたびにHLペアレジスタに1を加えて次の番地にしていくことにします．基本的なことは，**リスト2-1**のプログラムと変わりありません．

プログラムは**リスト2-6**のようになります．いかがですか？　これでモノクロモードにおいてのアトリビュートエリアの操作が理解できたと思います．いろいろ工夫して，プログラムを作るときに応用してみてください．

**表2-2　属性コード**

| ビット<br>表示する属性 ＼ 属性の意味 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 表示する属性コード |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | アンダライン | アッパライン | 0 | リバース | ブリンク | シークレット | |
| ブリンク | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 02H |
| リバース | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 04H |
| シークレット | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 01H |
| アッパライン | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 10H |
| アンダライン | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 20H |
| ブリンク・リバース | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 06H |
| アッパ・アンダライン | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 30H |
| アッパ・アンダライン，ブリンク | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 32H |

**●リスト2-6　8属性を1桁に表示**

```
                    ;********* Monochrome Text Mode ***********
                    ;
                    ; ﾚﾝｼｭｳ ﾉ 4-4 ( MIX )
B900                START: EQU  0B900H     ;START ADDRESS
E826                MON:   EQU  0E826H     ;MONITOR ADDRESS
F3C8                CRTAD: EQU  0F3C8H     ;V-RAM START ADDRESS
F418                SKETA: EQU  0F418H     ;V-RAM  AV ADDRESS
0002                AV1:   EQU  02H        ; BLINK DATA 02H
0004                AV2:   EQU  04H        ; REVERSE DATA 04H
0001                AV3:   EQU  01H        ; SECRET DATA 01H
0010                AV4:   EQU  10H        ; UPPER LINE DATA 10H
0020                AV5:   EQU  20H        ; UNDER LINE DATA 20H
0006                AV6:   EQU  06H        ; REVERSE AND BLINK DATA 06H
0030                AV7:   EQU  30H        ; UPPER & UNDER LINE DATA 30H
0032                AV8:   EQU  32H        ; UPPER & UNDER LINE & BLINK DATA
0000                CHG1:  EQU  00H        ; 0(DECIMAL DATA)
000A                CHG2:  EQU  0AH        ;10(DECIMAL DATA)
0014                CHG3:  EQU  14H        ;20(DECIMAL DATA)
001E                CHG4:  EQU  1EH        ;30(DECIMAL DATA)
0028                CHG5:  EQU  28H        ;40(DECIMAL DATA)
0032                CHG6:  EQU  32H        ;50(DECIMAL DATA)
003C                CHG7:  EQU  3CH        ;60(DECIMAL DATA)
0046                CHG8:  EQU  46H        ;70(DECIMAL DATA)
0041                CHAR:  EQU  41H        ;CHAR DATA 'A'
                    ;
                           ORG  START
                    ;
B900 21C8F3                LD   HL.CRTAD
B903 1650                  LD   D.50H      ;1 LINE DATA 80KETA
B905 3E41                  LD   A.CHAR     ;DISPLAY CHAR CODE
```

```
B907 77        LOOP1: LD   (HL),A
B908 23               INC  HL
B909 15               DEC  D
B90A C207B9           JP   NZ,LOOP1
B90D 2118F4           LD   HL,SKETA
B910 3E00             LD   A,CHG1
B912 77               LD   (HL),A
B913 23               INC  HL
B914 3E02             LD   A,AV1
B916 77               LD   (HL),A
B917 23               INC  HL
B918 3E0A             LD   A,CHG2
B91A 77               LD   (HL),A
B91B 23               INC  HL
B91C 3E04             LD   A,AV2
B91E 77               LD   (HL),A
B91F 23               INC  HL
B920 3E14             LD   A,CHG3
B922 77               LD   (HL),A
B923 23               INC  HL
B924 3E01             LD   A,AV3
B926 77               LD   (HL),A
B927 23               INC  HL
B928 3E1E             LD   A,CHG4
B92A 77               LD   (HL),A
B92B 23               INC  HL
B92C 3E10             LD   A,AV4
B92E 77               LD   (HL),A
B92F 23               INC  HL
B930 3E28             LD   A,CHG5
B932 77               LD   (HL),A
B933 23               INC  HL
B934 3E20             LD   A,AV5
B936 77               LD   (HL),A
B937 23               INC  HL
B938 3E32             LD   A,CHG6
B93A 77               LD   (HL),A
B93B 23               INC  HL
B93C 3E06             LD   A,AV6
B93E 77               LD   (HL),A
B93F 23               INC  HL
B940 3E3C             LD   A,CHG7
B942 77               LD   (HL),A
B943 23               INC  HL
B944 3E30             LD   A,AV7
B946 77               LD   (HL),A
B947 23               INC  HL
B948 3E46             LD   A,CHG8
B94A 77               LD   (HL),A
B94B 23               INC  HL
B94C 3E32             LD   A,AV8
B94E 77               LD   (HL),A
                ;
B94F C326E8           JP   MON              ;MONITOR
                ;
B952                  END
```

Section 6

# カラーモードで
# アトリビュートエリアを操作

## テキスト画面をカラー化する

PC-8801mkIISR は，スイッチを入れて BASIC が起動したときには，テキスト画面のモノクロモードに設定されています．BASIC でカラー画面にすることは簡単なのですが，マシン語ではそう簡単ではありません．というのは，PC-8801mkIISR は，「μPD3301」という CRT コントローラを使用しているために，DMA (Direct Memory Access) 操作をする必要があり，マシン語によるテキスト画面のカラー化は非常に大変です．そこで，今回は BASIC のコマンドをマシン語で実行させる方法を使用してみることにしました．

### ■ BASIC のコマンドをマシン語で実行させる

原理は簡単です．BASIC の

SCREEN, CONSOLE, DEF, USR

をマシン語で実行させれば良いのです．

すると，BASIC で打ち込まれた BASIC 命令はそのままメモリに記憶されずに，一旦中間言語に変換されてから，テキストウィンドウに入ります(**図2-10**)．この命令を ROM 内のサブルーチンで呼び出し，実行します．

具体的には，次の命令をBASICで打ち込みます．

図2-10　BASIC命令はテキストウィンドウへ

●リスト2-7　打ち込むBASIC命令

```
10 SCREEN 1:CONSOLE 0,25,0,1:DEF USR=&HB900:A=USR(0)
```

これからわかるように，カラーモードにしてからマシン語のプログラムのB900H番地へジャンプすることになっています．しかし，マシン語のプログラムではこのまま入力するのではなく，次のデータを8000H番地から入力します．

**●リスト2-8　入力するデータ**

```
8000 00 23 00 0A 00 D3 20 12 3A 9D 20 11 2C 0F 19 2C
8010 11 2C 12 3A 97 20 E0 F1 0C 00 B9 3A 41 F1 E0 28
8020 11 29 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

これは，先程のBASICプログラムが中間言語に直された結果のデータです．800EH番地が12Hでカラー画面に，11Hでモノクロ画面になります．また，マシン語の実行番地は801AHと8019H番地にアドレスを入れることによって，どこからでもスタートすることが可能です．もちろん，このままでは実行できませんので，次のサブルーチンを呼び出します．

JP　0B7CH

これを利用したプログラムを**リスト2-9**に示します．B920H番地からの中間言語を8000H番地に転送してから“JP　0B7CHを実行するようになっています．B932H番地を11Hにするとモノクロモードになります．また，ジャンプする番地はB939Hに下位バイト，B93AHに上位バイトを設定すれば，どこにでもジャンプすることができます．

**●リスト2-9　テキスト画面のカラー化**

```
                    ;***** TEXT COLOR ******
                    ;
                    ;TEXT ノ カラーカ ノ ルーチン
                    ;
B900                START: EQU  0B900H
E626                MON:   EQU  0E626H
B920                SRCE:  EQU  0B920H
8000                DEST:  EQU  8000H
0B7C                RUN:   EQU  0B7CH
                    ;
                           ORG  START
                    ;
B900 2120B9                LD   HL,SRCE
B903 110080                LD   DE,DEST
B906 012500                LD   BC,0025H
B909 7E             LOOP:  LD   A,(HL)   ┐
B90A 12                    LD   (DE),A   │
B90B 23                    INC  HL       │
B90C 13                    INC  DE       ├ 注
B90D 05                    DEC  B        │
B90E C209B9                JP   NZ,LOOP  ┘
B911 C37C0B                JP   RUN
                    ;
                    ;
```

```
B914                    DS    0CH ←── データの内容はともかく
          ;                          0CHバイト分確保する
          ;
B920 00                 DB    00H
B921 23                 DB    23H
B922 00                 DB    00H
B923 0A                 DB    0AH
B924 00                 DB    00H
B925 D3                 DB    0D3H
B926 20                 DB    20H
B927 12                 DB    12H
B928 3A                 DB    3AH
B929 9D                 DB    9DH
B92A 20                 DB    20H
B92B 11                 DB    11H
B92C 2C                 DB    2CH
B92D 0F                 DB    0FH
B92E 19                 DB    19H
B92F 2C                 DB    2CH
B930 11                 DB    11H
B931 2C                 DB    2CH
B932 12                 DB    12H          ; COLOR 12H,MONOCHROME 11H
B933 3A                 DB    3AH
B934 97                 DB    97H
B935 20                 DB    20H
B936 E0                 DB    0E0H
B937 F1                 DB    0F1H
B938 0C                 DB    0CH
B939 50                 DB    50H          ;PROGRAM LOW ADDRESS
B93A B9                 DB    0B9H         ;PROGRAM HIGH ADDRESS
B93B 3A                 DB    3AH
B93C 41                 DB    41H
B93D F1                 DB    0F1H
B93E E0                 DB    0E0H
B93F 28                 DB    28H
B940 11                 DB    11H
B941 29                 DB    29H
B942 00                 DB    00H
B943 00                 DB    00H
B944 00                 DB    00H
          ;
B945                    DS    0BH
          ;
B950 C326E6   MAIN:     JP    MON          ; MAIN PROGRAM
          ;
B953                    END
```

## ■ F419H 番地にデータを書き込んでテキストを 8 色に

Gコマンドを使用して B900H から実行してみてください．モニタのプロンプトが出てきました．このままでは本当にカラーモードになったのかわかりませんので，テストしてみましょう．

次のコマンドを実行させてください．

h] EF3C8 ⏎

Eコマンドで F3C8H 番地からの VRAM エリアのデータを表示させるものです．**リスト2-10**のように CRT 画面に VRAM のデータが表示されます．

●リスト2-10 "EF3C8"の実行結果

```
F3C0 FF 00 FF 00 FF 00 FF 00 46 33 43 30 20 46 46 20         F3C0 FF
F3D0 30 30 20 46 46 20 30 30 20 46 46 20 30 30 20 46    00 FF 00 FF 00 F
F3E0 46 20 30 30 20 34 36 20 33 33 20 34 33 20 33 30    F 00 46 33 43 30
F3F0 20 32 30 20 34 36 20 34 36 20 32 30 20 20 20 20     20 46 46 20
F400 20 20 20 FF 00 FF 00 FF 00 FF 00 46 33 43 30 20              F3C0
F410 46 46 20 20 20 20 20 20 80 E8 80 E8 80 E8 80 E8    FF        _♠_♠_♠_♠
F420 80 E8 80 E8 80 E8 80 E8 80 E8 80 E8 80 E8 80 E8    _♠_♠_♠_♠_♠_♠_♠_♠
F430 80 E8 80 E8 80 E8 80 E8 80 E8 80 E8 80 E8 80 E8    _♠_♠_♠_♠_♠_♠_♠_♠
F440 46 33 44 30 20 33 30 20 33 30 20 32 30 20 34 36    F3D0 30 30 20 46
F450 20 34 36 20 32 30 20 33 30 20 33 30 20 32 30 20     46 20 30 30 20
F460 34 36 20 34 36 20 32 30 20 33 30 20 33 30 20 32    46 46 20 30 30 2
F470 30 20 34 36 20 20 20 20 20 20 20 30 30 20 46 46    0 46        00 FF
F480 20 30 30 20 46 46 20 30 30 20 46 20 20 20 20 20     00 FF 00 F
F490 80 E8 80 E8 80 E8 80 E8 80 E8 80 E8 80 E8 80 E8    _♠_♠_♠_♠_♠_♠_♠_♠
F4A0 80 E8 80 E8 80 E8 80 E8 80 E8 80 E8 80 E8 80 E8    _♠_♠_♠_♠_♠_♠_♠_♠
F4B0 80 E8 80 E8 80 E8 80 E8 46 33 45 30 20 34 36 20    _♠_♠_♠_♠F3E0 46
```

1行目のアトリビュートエリアの番地はF418H番地から始まります．そこで，F419H番地の属性コードにカラーデータを書き込みます．それで1行目はカラー文字になるはずです．カラーの属性データは**表2-3**のようになります．

F419H番地にこの属性コードを書き込めば，任意の色を出すことができます．もちろん，モノクロモードでやったように，1行の文字を何色かに色分けすることもできます．

**表2-3 カラーの属性コード**

| コード | 色 |
|---|---|
| 08H | 黒 |
| 28H | 青 |
| 48H | 赤 |
| 68H | 紫 |
| 88H | 緑 |
| A8H | 水色 |
| C8H | 黄色 |
| E8H | 白 |

## サブルーチンを使ったテキストのカラー化

今度は，カラー化で一般的であるサブルーチンを使用して，カラーモードにしてみましょう．しかし，この方法では，バックグランドのコントロールはできません．この方法は，前の方法と比べるとプログラム自体は短くなりますが，バックグランドをコントロールできないのが残念です．

画面コントロール，エントリ・ポイント　6F6BH

これをただCALLするだけではだめで，数個のパラメータが必要です．エントリ・ポイントをCALLする前に，パラメータをセットしなければなりません．パラメータは**表2-4**(P.96)のようになっていますので，じっくり見てください．また，プログラムは**リスト2-11**(P.96)となります．

**表2-4　画面コントロール（6F6BH）のパラメータ**

| | |
|---|---|
| Bレジスタ | 画面の桁数（1～80桁） |
| Cレジスタ | 画面の行数（1～25行） |
| E6B2H | スクロール開始行 |
| E6B3H | スクロール終了行 |
| E6B4H | カラーコード |
| E6B8H | ファンクションキー表示フラグ（0以外で表示） |
| E6B9H | テキスト・カラーマーク（FFHでカラー，00Hでモノクロ） |

E6B4Hにカラーコードをセットすれば任意の色を出すことができます．また，EコマンドでF3C8Hから表示させて，F419H番地にカラーデータを入力すると，1行目の文字に色がつきます．カラーコードは(P.95)と同じです．

**●リスト2-11　テキスト画面のカラー化（その2）**

```
                    ;***** COLOR TEXT MODE *****
                    ;
                    ; TEXT ｶﾗｰｶ ﾉ ﾙｰﾁﾝ (CALL 6F6BH)
                    ;
B900                START: EQU  0B900H
E626                MON:   EQU  0E626H        ;ﾓﾆﾀｰ ﾉ ｴﾝﾄﾘｰ ﾎﾟｲﾝﾄ
6F6B                COLOR: EQU  6F6BH         ;ｶﾗｰ ﾓｰﾄﾞ ﾙｰﾁﾝ
                    ;
                           ORG  START
                    ;
B900 0650                  LD   B,50H         ;CRT ﾉ ｹﾀｽｳ  ｦ ｾｯﾄ
B902 0E19                  LD   C,19H         ;CRT ﾉ ｷﾞｮｳｽｳ ｦ ｾｯﾄ
B904 3E01                  LD   A,01H         ;ｽｸﾛｰﾙ ﾉ ｶｲｼ ﾉ ｷﾞｮｳ ｦ ｾｯﾄ
B906 32B2E6                LD   (0E6B2H),A
B909 3E19                  LD   A,19H         ;ｽｸﾛｰﾙ ﾉ ｵﾜﾘ ﾉ ｷﾞｮｳ ｦ ｾｯﾄ
B90B 32B3E6                LD   (0E6B3H),A
B90E 3EE8                  LD   A,0E8H        ;ｶﾗｰｺｰﾄﾞ ｦ ｾｯﾄ
B910 32B4E6                LD   (0E6B4H),A
B913 3E00                  LD   A,00H         ;ﾌｧﾝｸｼｮﾝ ｷｰ ﾉ ﾋｮｳｼﾞ ﾉ ON OFF
B915 32B8E6                LD   (0E6B8H),A
B918 3EFF                  LD   A,0FFH        ;ｶﾗｰﾏｰｸ ﾉ ｾｯﾄ ｶﾗｰﾏｰｸ FFH
B91A 32B9E6                LD   (0E6B9H),A
                    ;
B91D CD6B6F                CALL COLOR
                    ;
B920 C326E6                JP   MON
                    ;
B923                       END
```

## バッググランド・カラーを変える

キャラクタ単位のカラー化はできましたので，今度はバックグランドのカラー化について考えてみましょう．

バックグランドのカラーコントロールは，I/O ポートの 52H が行うようになっています．I/O ポートの 52H にカラーデータを入力すればいいのです．カラーデータは**表2-5**のようになっています．

このテストはモニタコマンドを使用することによって簡単にできます．"O"コマンドを使用すればよいのです．ですから，たとえば赤を表示したいのであれば，

```
h] O52, 20 ⏎
        └────赤のデータ
```

とすればバックグランドの色は赤に変ります．いろいろなデータで確めてください．なお，バックグランドの色が白になった場合は，表示されているデータは読むことができません．これは，キャラクタの色とバックグランドの色が同じになるためです．

**表2-5　バックグランドのカラーデータ**

I/Oポートアドレス　52H

| データ | 色 |
|---|---|
| 00H | 黒 |
| 10H | 青 |
| 20H | 赤 |
| 30H | 紫 |
| 40H | 緑 |
| 50H | 水色 |
| 60H | 黄色 |
| 70H | 白 |

## ブロック転送命令を使う

**リスト2-9**のプログラムではB920H番地からのデータを8000H番地に転送しています．この転送のために使用される命令は，全部で9命令ですが，4命令にすることができます．

Z-80には，ブロック転送命令という強力な命令があります．このブロック転送命令は6命令を1命令にしたようなもので，なおかつ6命令を使用するより速く処理できるものです．この命令は次のようになります．

LDIR

書式　LDIR

機能　HL ペアレジスタで指定される番地のデータを DE ペアレジスタで示される番地に，BC ペアレジスタが 0 になるまで転送します．

```
例      LD  HL, 0B920H      (転送元のアドレスをセットする)
        LD  DE, 8000H       (転送先のアドレスをセットする)
        LD  BC, 0025H       (転送するバイト数をセットする)
        LDIR
```

## ■ LDIR を使った場合と使わない場合を比較すると

LDIR 命令は，実行する前に各ペアレジスタにデータをセットしてからでないと実行することはできませんが強力な命令です．ためしに，この命令を使用した場合と使用していない場合を比べてみましょう．

```
   LDIRを使わなかった場合         LDIRを使った場合
       LD  HL, 0B920H              LD  HL, 0B920H ┐
       LD  DE, 8000H               LD  DE, 8000H  ├ 共通
       LD  BC, 0025H               LD  BC, 0025H  ┘

LOOP : LD  A, (HL)                  LDIR
       LD  (DE), A
       INC  HL
       INC  DE
       DEC  B
       JP  NZ, LOOP
```

と，このようになります．これだったら，使用しないと損ですね．どんどん使っていくことにしましょう．

このLDIR 命令を使用したプログラムを**リスト2-12**に示します．**リスト2-9**と動作内容は全て同じです．

**●リスト2-12　LDIRを使ったテキストカラー化**

```
                ;***** TEXT COLOR ******
                ;
                ;TEXT ノ カラーカ ノ ルーチン
                ;
B900            START:  EQU   0B900H
E626            MON:    EQU   0E626H
B920            SRCE:   EQU   0B920H
8000            DEST:   EQU   8000H
0B7C            RUN:    EQU   0B7CH
                ;
                        ORG   START
                ;
```

```
B900 2120B9            LD    HL,SRCE
B903 110080            LD    DE,DEST
B906 012500            LD    BC,0025H
B909 EDB0              LDIR
B90B C37C0B            JP    RUN
              ;
              ;
B90E                   DS    12H
              ;
              ;
B920 00                DB    00H
B921 23                DB    23H
B922 00                DB    00H
B923 0A                DB    0AH
B924 00                DB    00H
B925 D3                DB    0D3H
B926 20                DB    20H
B927 12                DB    12H
B928 3A                DB    3AH
B929 9D                DB    9DH
B92A 20                DB    20H
B92B 11                DB    11H
B92C 2C                DB    2CH
B92D 0F                DB    0FH
B92E 19                DB    19H
B92F 2C                DB    2CH
B930 11                DB    11H
B931 2C                DB    2CH
B932 12                DB    12H          ; COLOR 12H,MONOCHROME 11H
B933 3A                DB    3AH
B934 97                DB    97H
B935 20                DB    20H
B936 E0                DB    0E0H
B937 F1                DB    0F1H
B938 0C                DB    0CH
B939 50                DB    50H          ;PROGRAM LOW ADDRESS
B93A B9                DB    0B9H         ;PROGRAM HIGH ADDRESS
B93B 3A                DB    3AH
B93C 41                DB    41H
B93D F1                DB    0F1H
B93E E0                DB    0E0H
B93F 28                DB    28H
B940 11                DB    11H
B941 29                DB    29H
B942 00                DB    00H
B943 00                DB    00H
B944 00                DB    00H
B945                   DS    0BH
              ;
B950 C326E6   MAIN:    JP    MON                ; MAIN PROGRAM
              ;
B953                   END
```

*Section 7*

# TEXT文字とバックグランドのカラー化デモプログラム

## デモプログラムの仕様

TEXT 文字とバックグランドカラーのカラー化が理解できたと思いますので，仕上げの意味も含めて，カラーのデモプログラムを作ってみましょう．

**図2-11**のような画面のレイアウトを考えていきましょう．まず，プログラムの仕様をはっきり決めておかなければなりません．プログラムを作る前に仕様を決めておかないと，初めの目的がわからなくなってしまうからです．

プログラムの仕様

a．同一キャラクタを画面全部に表示させる（80×25モード）

b．キャラクタを“→”（1CH）～“秒”（F7H）まで変化させる

c．キャラクタの色を順次に8色表示させる

d．キャラクタの色1色に対して，バックグランドの色を順次8色にする

e．全部表示し終ったら，モニタに戻る

大まかなフローチャートを考えると**図2-12**のようになります．

**図2-11　画面のレイアウト**

キャラクタを“→”(1CH)から“秒”(F7H)まで変化させる

80桁

25行

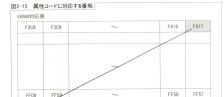
文字の色を順次8色に変えていく

バックグランドの色を順次8色にする

図2-12 ラフ・フローチャート

# 全体のキャラクタに色をつけるには

CRT全体に色をつけるためには，アトリビュートエリアの属性コードに対応する番地に，属性コードを書き込む必要があります．付録のVRAM対応表を見てください．これには，アトリビュートエリアの対応表はありませんが，属性コードに対応する番地はすぐわかります．**図2-13**のようになります．

**図2-13　属性コードに対応する番地**

●この番地に＋1すると桁数の指定番地．以後は2H番地とばす
　例　F418H　1回目の桁数の指定番地
　　　F41AH　2回目の桁数の指定番地

●＋2すると属性コードの指定番地．以後は2H番地とばす
　例　F419H　1回目の属性の指定番地
　　　F41BH　2回目の属性の指定番地

したがって，1桁ごとの属性コード指定番地へ属性コードを書き込まなければなりませんので，全部で25回の作業が必要となります．しかし，下の行までの番地は120バイト分あるので，アドレスに78H(120)分を加えれば済みます．

## ■カラーの属性コードは20Hとび

カラーの属性コードは08H～E8Hまで変化するのですが，都合悪いことに連続になっておらず，20Hとびになっています．したがって，初めのデータに20H加えて属性コードを変えてやらなければなりません．

例　初めのデータ　08H
　　2回目のデータ　28H　(08H＋20H)
　　3回目のデータ　48H　(28H＋20H)
　　　⋮
　　8回目のデータ　E8H　(C8H＋20H)

## バックグランドに色をつけるには

プログラムでバックグランドに色をつける方法は，I/O ポートの52H へカラーデータを送ればよいのです．カラーコードは00H～70H まで変化しますが，これも10H とびになっていますので，初めのデータ00H に10H ずつ足していけばよいことになります．

例　初めのデータ　00H
　　2回目のデータ　10H　(00H＋10H)
　　3回目のデータ　20H　(10H＋10H)
　　　⋮
　　8回目のデータ　70H　(60H＋10H)

### ■旧8801ならばボーダカラーも操作できる

ただし，旧 PC-8801をお持ちの方は，ボーダカラーも操作することができます．下位3ビットがボーダカラーのビットですので，この3ビットに0～7 H までのデータを書き込みます．

```
例    11H
      ||
      |└──────ボーダカラーのデータ（青色になる）
      └───────バックカラーのデータ（青色）
```

この例では，CRT の画面全体が青色になります．なお，mkII や SR では残念ながら使用できません．

## プログラムはこうなっている

では，**リスト2-13**にテキスト画面のカラー化デモプログラムを示します．これから，プログラムリストの説明をしていきましょう．

**●リスト2-13　テキスト画面のカラー化デモプログラム**

```
                    ;***** TEXT COLOR ******
                    ;
                    ;PROGRAM NAME --> ctdemo.e
                    ;
                    ;TEXT ノ ｶﾗｰﾉ ﾃﾞﾓﾌﾟﾛｸﾞﾗﾑ
                    ;
 B900               START: EQU  0B900H
 E626               MON:   EQU  0E626H
 B920               SRCE:  EQU  0B920H
 8000               DEST:  EQU  8000H
 0B7C               RUN:   EQU  0B7CH
 F3C8               ADCRT: EQU  0F3C8H
                    ;
```

```
                        ORG   START
                    ;
                    ;CRT MODE PROGRAM -------------------------
                    ;
B900  2120B9            LD    HL,SRCE
B903  110080            LD    DE,DEST
B906  012500            LD    BC,0025H
B909  EDB0              LDIR
B90B  C37C0B            JP    RUN
                    ;
                    ;
B90E                    DS    12H
                    ;
                    ;
B920  00                DB    00H
B921  23                DB    23H
B922  00                DB    00H
B923  0A                DB    0AH
B924  00                DB    00H
B925  D3                DB    0D3H
B926  20                DB    20H
B927  12                DB    12H
B928  3A                DB    3AH
B929  9D                DB    9DH
B92A  20                DB    20H
B92B  11                DB    11H
B92C  2C                DB    2CH
B92D  0F                DB    0FH
B92E  19                DB    19H
B92F  2C                DB    2CH
B930  11                DB    11H
B931  2C                DB    2CH
B932  12                DB    12H           ; COLOR 12H,MONOCHROME 11H
B933  3A                DB    3AH
B934  97                DB    97H
B935  20                DB    20H
B936  E0                DB    0E0H
B937  F1                DB    0F1H
B938  0C                DB    0CH
B939  50                DB    50H           ;MAIN PROGRAM LOW ADDRESS
B93A  B9                DB    0B9H          ;MAIN PROGRAM HIGH ADDRESS
B93B  3A                DB    3AH
B93C  41                DB    41H
B93D  F1                DB    0F1H
B93E  E0                DB    0E0H
B93F  28                DB    28H
B940  11                DB    11H
B941  29                DB    29H
B942  00                DB    00H
B943  00                DB    00H
B944  00                DB    00H
                    ;
B945                    DS    0BH
                    ;
                    ; MAIN PROGRAM--------------------------
                    ;
B950  CD8B42        MAIN:  CALL  428BH          ;CURSOR OFF
B953  3E1C              LD    A,1CH            ;CHR START DATA
B955  324DB9            LD    (0B94DH),A       ;CHR DATA AREA
B958  3E08              LD    A,08H
                    ;
```

```
B95A CDA7B9   SMAIN: CALL DISP
B95D 0608            LD   B,08H
B95F 2119F4   CCOLR: LD   HL,0F419H
B962 117800          LD   DE,0078H
B965 0E19            LD   C,19H
B967 77       CLOP1: LD   (HL),A
B968 19              ADD  HL,DE
B969 0D              DEC  C
B96A C267B9          JP   NZ,CLOP1
              ;
B96D C5       BCOLR: PUSH BC
B96E F5              PUSH AF
B96F 3E00            LD   A,00H
B971 0610            LD   B,10H
B973 0E08            LD   C,08H
B975 D352     BLOP1: OUT  (52H),A
B977 CDCCB9          CALL WAIT
B97A 80              ADD  A,B
B97B 0D              DEC  C
B97C C275B9          JP   NZ,BLOP1
B97F F1              POP  AF
B980 C1              POP  BC
B981 C5              PUSH BC
B982 0620            LD   B,20H
B984 80              ADD  A,B
B985 C1              POP  BC
B986 05              DEC  B
B987 C25FB9          JP   NZ,CCOLR
B98A F5              PUSH AF
B98B 3A4DB9          LD   A,(0B94DH)
B98E FEF8            CP   0F8H
B990 C2A3B9          JP   NZ,SMJ
              ;
B993 CD9042          CALL 4290H               ;CURSOR ON
              ;
B996 3E00            LD   A,00H               ;BACK COLOR TO BLACK
B998 D352            OUT  (52H),A
              ;
B99A 3E0C            LD   A,0CH               ;CRT CLEAR DATA 0CH
B99C CD1800          CALL 0018H               ;CRT CLEAR SUB
B99F F1              POP  AF
              ;
B9A0 C326E6          JP   MON
              ;
B9A3 F1       SMJ:   POP  AF
B9A4 C35AB9          JP   SMAIN
              ;
              ;DISP SUB -----------------
              ;
B9A7 E5       DISP:  PUSH HL
B9A8 D5              PUSH DE
B9A9 C5              PUSH BC
B9AA F5              PUSH AF
B9AB 21C8F3          LD   HL,ADCRT
B9AE 112800          LD   DE,0028H
B9B1 0E19            LD   C,19H
B9B3 0650     DLOP2: LD   B,50H
B9B5 3A4DB9          LD   A,(0B94DH)
```

```
B9B8 77         DLOP1: LD    (HL),A
B9B9 23                INC   HL
B9BA 05                DEC   B
B9BB C2B8B9            JP    NZ,DLOP1
B9BE 19                ADD   HL,DE
B9BF 0D                DEC   C
B9C0 C2B3B9            JP    NZ,DLOP2
B9C3 3C                INC   A
B9C4 324DB9            LD    (0B94DH),A
B9C7 F1                POP   AF
B9C8 C1                POP   BC
B9C9 D1                POP   DE
B9CA E1                POP   HL
B9CB C9                RET
                ;
                ;WAIT SUB -------------------
                ;
B9CC C5         WAIT:  PUSH  BC
B9CD F5                PUSH  AF
B9CE 0E01              LD    C,01H
B9D0 06A0       WL3:   LD    B,0A0H
B9D2 3EFF       WL2:   LD    A,0FFH
B9D4 3D         WL1:   DEC   A
B9D5 C2D4B9            JP    NZ,WL1
B9D8 05                DEC   B
B9D9 C2D2B9            JP    NZ,WL2
B9DC 0D                DEC   C
B9DD C2D0B9            JP    NZ,WL3
B9E0 F1                POP   AF
B9E1 C1                POP   BC
B9E2 C9                RET
                ;
B9E3                   END
```

## ■ここで使用しているROM内ルーチン

CRTのモード設定が終った後のB950H番地から，メインプログラムに入っていますが，少し簡単にするためにROM内ルーチンを使用しています．

ROM内ルーチンは次のようになります．

| ROMの番地 | サブルーチンの意味 |
|---|---|
| 428BH | カーソル表示を消す |
| 4290H | カーソルを表示する |
| 0018H | CRTをクリアする．ただし，AレジスタにOCHを入れる必要がある |

## ■メインプログラム内での汎用レジスタの用途

B94DH番地は，キャラクタコードをストアしておくワークエリアになります．これはレジスタがすべて使用されているために必要とするものです．

次に，メインプログラム内での汎用レジスタの用途について述べることにします．

| | |
|---|---|
| HL ペアレジスタ | アトリビュートエリアの属性コード番地 |
| DE ペアレジスタ | 真下の行に持ってくるためのデータ(120バイト分) |
| B レジスタ | キャラクタの色の個数 |
| C レジスタ | 行数を数える |
| A レジスタ | カラーコードデータ |

これ以外の用途で使用される場合は，すべてPUSH命令で退避させています．

最後のキャラクタコードのF7Hの表示が終ったならば，カーソルを表示して，バックグランドの色を黒にしています．というのは，最後のバックグランドの色が白になると，プロンプトがどこにあるかわからなくなるからです．最後に画面をクリアして，モニタに戻ります．

このプログラムは，終了するまでに約1時間かかります．“WAIT”のサブルーチンのデータを変えれば、もう少し速くなりますが，速くなりすぎてバックグランドの色を変化させるときにチラツキが表れてしまいます．もし，速くしたい人は，B9D1H番地のデータA0Hを変えて小さい値にしてください．

## ■参考―テキスト画面のカラー化デモプログラム　その2

このプログラムは，「画面コントロール・サブルーチン」(**リスト2-13**)を使用したもので，**リスト2-13**と同じように作ってあります．ただし，バックグランドのカラー化はできないので省いてあります．また，WAIT時間(表示している時間)は長くしないとはっきり見えませんので，少し，長くしてあります．

**●リスト2-14 テキスト画面のカラー化デモプログラム その2**

```
                    ;***** TEXT COLOR ******
                    ;
                    ;PROGRAM NAME --> csdemo.e
                    ;
                    ;TEXT ﾉ ｶﾗｰﾉ ﾃﾞﾓﾌﾟﾛｸﾞﾗﾑ ｿﾉ 2 sub( CALL 6F6BH)
                    ;
B900                START: EQU  0B900H
E626                MON:   EQU  0E626H
F3C8                ADCRT: EQU  0F3C8H
6F6B                COLOR: EQU  6F6BH              ;CRT COLOR SUBROUTINE
                    ;
                    ;
                    ;
                    ;
                           ORG  START
                    ;
                    ;CRT MODE PROGRAM -------------------------
                    ;
B900 0650                  LD   B,50H   ;ﾖｺ ﾉ ｹﾀｽｳ ｦ ｾｯﾄ ﾄﾙ 0<Bﾚｼﾞｽﾀ>80
B902 0E19                  LD   C,19H   ;ﾀﾃ ﾉ ｷﾞｮｳ ｽｳ ｦ ｾｯﾄ ｽﾙ 0<Cﾚｼﾞｽﾀ>25
B904 3E01                  LD   A,01H   ;ｽｸﾛｰﾙ ｶｲｼ ｷﾞｮｳ
```

```
B906 32B2E6           LD    (0E6B2H),A
B909 3E19             LD    A,19H          ;スクロール シュウリョウ ギョウ
B90B 32B3E6           LD    (0E6B3H),A
B90E 3EE8             LD    A,0E8H      ;カラーコード
B910 32B4E6           LD    (0E6B4H),A
B913 3E00             LD    A,00H       ;ファンクション コード ON,01H OFF,00H
B915 32B8E6           LD    (0E6B8H),A
B918 3EFF             LD    A,0FFH      ;カラー マーク カラー,FFH モノクロ,00H
B91A 32B9E6           LD    (0E6B9H),A
B91D CD6B6F           CALL  COLOR
                ;
B920                  DS    01H            ;CHR DATA AREA
                ;
                ; MAIN PROGRAM--------------------------
                ;
B921 CD8B42     MAIN: CALL  428BH          ;CURSOR OFF
B924 3E1C             LD    A,1CH          ;CHR START DATA
B926 3220B9           LD    (0B920H),A     ;CHR DATA AREA
B929 3E08             LD    A,08H
                ;
B92B CD63B9     SMAIN: CALL DISP
B92E 0608             LD    B,08H
B930 2119F4     CCOLR: LD   HL,0F419H
B933 117800           LD    DE,0078H
B936 0E19             LD    C,19H
B938 77         CLOP1: LD   (HL),A
B939 19               ADD   HL,DE
B93A 0D               DEC   C
B93B C238B9           JP    NZ,CLOP1
B93E CD88B9           CALL  WAIT
B941 C5               PUSH  BC
B942 0620             LD    B,20H
B944 80               ADD   A,B
B945 C1               POP   BC
B946 05               DEC   B
B947 C230B9           JP    NZ,CCOLR
B94A F5               PUSH  AF
B94B 3A20B9           LD    A,(0B920H)
B94E FEF8             CP    0F8H
B950 C25FB9           JP    NZ,SMJ
                ;
B953 CD9042           CALL  4290H          ;CURSOR ON
                ;
                ;
B956 3E0C             LD    A,0CH          ;CRT CLEAR DATA 0CH
B958 CD1800           CALL  0018H          ;CRT CLEAR SUB
B95B F1               POP   AF
                ;
B95C C326E6           JP    MON
                ;
B95F F1         SMJ:  POP   AF
B960 C32BB9           JP    SMAIN
                ;
                ;DISP SUB ----------------
                ;
B963 E5         DISP: PUSH  HL
B964 D5               PUSH  DE
B965 C5               PUSH  BC
B966 F5               PUSH  AF
B967 21C8F3           LD    HL,ADCRT
B96A 112800           LD    DE,0028H
```

```
B96D 0E19              LD    C,19H
B96F 0650      DLOP2:  LD    B,50H
B971 3A20B9            LD    A,(0B920H)
B974 77        DLOP1:  LD    (HL),A
B975 23                INC   HL
B976 05                DEC   B
B977 C274B9            JP    NZ,DLOP1
B97A 19                ADD   HL,DE
B97B 0D                DEC   C
B97C C26FB9            JP    NZ,DLOP2
B97F 3C                INC   A
B980 3220B9            LD    (0B920H),A
B983 F1                POP   AF
B984 C1                POP   BC
B985 D1                POP   DE
B986 E1                POP   HL
B987 C9                RET
               ;
               ;WAIT SUB -------------------
               ;
B988 C5        WAIT:   PUSH  BC
B989 F5                PUSH  AF
B98A 0E02              LD    C,02H
B98C 06FF      WL3:    LD    B,0FFH
B98E 3EFF      WL2:    LD    A,0FFH
B990 3D        WL1:    DEC   A
B991 C290B9            JP    NZ,WL1
B994 05                DEC   B
B995 C28EB9            JP    NZ,WL2
B998 0D                DEC   C
B999 C28CB9            JP    NZ,WL3
B99C F1                POP   AF
B99D C1                POP   BC
B99E C9                RET
               ;
B99F                   END
```

2章ではテキスト画面とバックグランドをカラー化する方法を中心に話を進めてきましたが，いかがでしたか？　テキスト画面をカラー化する方法はいろいろあるのですが，これが一番簡単だと思います．テキスト画面をカラー化していろいろと楽しんでください．

# 第3章 キーボードを操作する

## キーボードからデータを読み取る

キャラクタ単位のカラー化ができましたので，この章ではキーボードからデータを読み取る方法を考えてみましょう．

PC-8801mkIISRでは，8ビットデータの各1ビットごとに1個のキーが割り当てられているので，キーが押されたかどうかが簡単にわかります．これを利用して，キー入力によってキャラクタを4方向，8方向に移動させるプログラムを作ってみましょう．

また，ちょっと耳慣れない「論理演算命令」についても勉強しましょう．この命令を使うとドット単位で画面を動かすこともできるので，キャラクタをスムーズに移動でき，大変便利です．

*Section 1*

# キーボードetc.に使われるI/O命令

## I/Oポートから8ビットデータをとり込むIN命令

I/Oとはよく使用される言葉です．I/OはInput／Outputの略であることは，御存知のことと思います．I/Oは本来，メモリ以外へ入出力することが目的となっていました．しかし，今はI/Oを利用してメモリバンクなどによる増設等，いろいろと利用されています．したがって，パソコンをうまく利用するためには，I/O命令を利用してI/Oポートをアクセスする必要があります．Z-80では，I/Oポートは全部でFFH (256) まで利用できます．まず，IN命令について説明していきましょう．

IN命令は，I/Oポートから8ビットデータを取り込む命令です．キーボードからデータを取り込んだり，フロッピィディスクからデータを取り込んだりするのに多く使用されます．

IN　A,（N）

書　式　　IN　A,（8ビットのポート番号）

機　能　　8ビットポート番号で指定されたポートから，Aレジスタにデータを取り込みます．

IN　A,（00H）

ポート番号00HからAレジスタにデータを取り込みます．

### ■ IN命令（Cレジスタ間接）

もうひとつIN命令があります．それはCレジスタ間接と言われるもので，Cレジスタのデータを I/Oポート番号と見なして実行するものです．Cレジスタは演算できますので，ポート番号が連続しているときなどは，非常に便利です．また，取り込むレジスタは，Aレジスタばかりでなくすべての汎用レジスタが

利用できます．

| IN　r，(C) | |
|---|---|
| 書　式 | IN　r，(C)（rはA，B，C，D，E，H，Lレジスタ） |
| 機　能 | Cレジスタで指定されるI/Oポートから，rで指定されるレジスタにデータを取り込みます． |
| 例 | LD　C，01H<br>IN　D，(C)<br>ポート番号の01Hから，Dレジスタにデータを取り込みます． |
| フラグ | フラグはデータの値により，次の3つが変化します．<br>P/Vフラグ：入力データの入っているビットが偶数のとき1，奇数のとき0<br>Sフラグ：入力データの最上位ビット(7ビット目)が1のとき1，0のとき0<br>Zフラグ：入力データが0のとき1，それ以外のとき0 |

この命令は，どのキーボードが押されたかを調べるときに使用されています．

## I/Oポートにデータを出力するOUT命令

OUT命令はIN命令とは逆に，I/Oポートにデータを出力する命令です．しかし，この命令はデータを無視して，I/Oポートのスイッチとして使用されることもあります．PC-8801mkIISRでは，カラーパレットやRGBのカラープレーンなどに利用されています．

| OUT　(n)，A | |
|---|---|
| 書　式 | OUT（ポート番号），A |
| 機　能 | ポート番号で指定されたポートに，Aレジスタのデータを出力します． |
| 例 | OUT（52H），A<br>ポート番号の52HへAレジスタのデータを出力します． |

OUT 命令にも，IN 命令と同様にCレジスタを使用した間接命令があります．

## ■ OUT 命令（C レジスタ間接）

OUT (C), r

書　式　OUT　(C), r（r は A, B, C, D, E, H, L レジスタ）
機　能　ポート番号で指定されたポートに，r で指定されるレジスタのデータを出力します．
例　　　LD　C, 52H
　　　　OUT (C), B
　　　　ポート番号の52Hへ B レジスタのデータを出力します．

なお，この命令では，フラグは変化しません．

I/O 命令はキーボードからの取り込みなどに多く使用されているので，応用のし方をしっかり覚えましょう．

*Section 2*

# キーボードから データを読みとるには

## ビットを調べればキー入力がわかる

PC-8801mkIISR は CPU(Z-80A) でキーボードを1個1個調べているので，キーボードを調べるプログラムをマシン語で作るのは比較的楽にできます．キーボードのキーの数は全部で90個ありますが，キーを8個ずつのデータに分けて，8ビットのデータとしてI/O命令で読み取ります．全部のキーを調べるためには，I/O命令を12回（0BH）使用しなければなりませんが，実際のプログラムでは，調べるキーの数は多くても10～20くらいでしょうから，I/O命令を多く使用することはないでしょう．SRでは，8ビットデータの各1ビットごとに，1個のキーが割り当てられています（図3-1，P117）．

### ■各ビットの状態とキーの状態

各ビットを調べれば，8個のキーの内どのキーが押されたかがわかります．キーの状態によって各ビットは次のように変化します．

| 各ビットの状態 | キーの状態 |
|---|---|
| 1 | 押されていない |

図3-2　キーボード・データとビットの対応関係

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| I/O アドレス | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 00H | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 01H | ↲ | . | , | = | + | * | 9 | 8 |
| ⋮ | | | | | | | | |
| 0BH | ROLL UP | ROLL DOWN | | | | | | |

0　　　　　押されている

押されているときが0の状態で，押されていないときが1の状態になっています．少し見にくいのですが，ソフトで何とでも変更できるので，気にする必要はありません．

図3-1　キーボード・データ

| I/Oアドレス ⇩ ＼ ビット⇨ | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 00H | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 01H | ↵ | ・ | , | = | + | * | 9 | 8 |
| 02H | G キ | F ハ | E イィ | D シ | C ソ | B コ | A チ | @ ` |
| 03H | O ラ | N ミ | M モ | L リ | K ノ | J マ | I ニ | H ク |
| 04H | W テ | V ヒ | U ナ | T カ | S ト | R ス | Q タ | P セ |
| 05H | －＝ホ | －＾へ | ］｝」ム | ¥｜－ | ［｛「 | Z ツッ | Y ン | X サ |
| 06H | 7 ’ ヤ ャ | 6 & オ ォ | 5 % エ ェ | 4 $ ウ ゥ | 3 # ア ァ | 2 ” フ | 1 ! ヌ | 0 ワ ヲ |
| 07H | _ ～ ロ | ／ ? メ ・ | . ＞ ル 。 | , ＜ ネ 、 | ; ＋ レ | : * ケ | 9 ) ヨ ョ | 8 ( ユ ュ |
| 08H | CTRL | SHIFT | カナ | GRPH | INS DEL | → | ↑ | HOME CLR |
| 09H | ESC | SPACE | f. 5 | f. 4 | f. 3 | f. 2 | f. 1 | STOP |
| 0AH | CAPS LOCK | ／ | － | COPY | HELP | ← | ↓ | TAB |
| 0BH | ROLL UP | ROLL DOWN | | | | | | |

# 対応ビットが0なら，キーは押された

**図3-1**を見ながら具体的に説明していくことにしましょう．キーが押されていないときはどのデータのビットも，すべて1になります．つまり，8ビットデータはFFHになります．

もし，どこかのキーが押されていれば，いずれかのビットが0になっています．つまり，FFH以外のデータになるわけです．

したがって，目的のキーが押されたかどうかを調べるには，目的のキーのあるI/Oアドレスを指定して，そのキーに対応するビットが0になっているかを調べれば良いわけです．具体的に表すと，**図3-3**のようになります．

**図3-3　I/Oアドレス00Hのときの具体的なデータ例**

| | ビット | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| I/Oアドレス00Hのとき | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| キーが押されていないとき | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | FFH |
| 5のキーが押されたとき | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | DFH |
| 1のキーが押されたとき | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | FDH |
| 1と3のキーが押されたとき | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | F5H |

## ■ Aのキーが押されたかを調べる

たとえば，“A”のキーが押されたかどうかを調べるプログラムを作ってみましょう．**図3-1**をもう一度見てください．“A”のキーはI/Oアドレス「02H」の1ビット目にあります．したがって，このビットが0であれば“A”のキーが押されたことになります．プログラムは次のようになります．

```
KEY ; IN   A, (02H)        ; Aのキーのある I/O アドレス
      CP   0FDH            ; Aのキーのデータ  11111101B
      JP   Z, MON          ; Aのキーが押されたらモニタへ戻る
      CALL WAIT            ; 時間待ちのサブルーチンを呼ぶ
      JP   KEY
```

```
WAIT ：LD   B   0FFH          ；時間待ちのサブルーチン
LOP  ：DEC  B
       JP   NZ, LOP
       RET
```

このように，簡単なプログラムで済みます．

### ■Dのキーを調べる際の変更点

もし，“A”のキーから，“D”のキーに変更するのであれば，データを次のように変えれば良いことになります．

| Aのデータ | Dのデータ |
|---|---|
| 11111101 B | 11101111 B |
| (FDH) | (EFH) |

つまり，FDH を EFH にするだけです．

## キー入力のテストプログラムを作る

キー入力のテストとして，スペースキーが押されたらモニタへ戻るプログラムを作ってみましょう．ラフ・フローチャートを図3-4に示します．

図3-4 ラフ・フローチャート

プログラムは，**リスト3-1**のようになります．新しい命令は使っていませんので，理解出来ると思います．

**●リスト3-1　キー入力のテスト**

```
                         ;
                         ;********* KEY IN TEST **************
                         ;
                         ;PROGRAM NAME --->keyt1.e
                         ;
                         ;ｷｰ ﾆｭｳﾘｮｸ ﾉ ﾃｽﾄ ﾌﾟﾛｸﾞﾗﾑ ( CP 0BFH )
                         ;
                         ;***********************************
                         ;
E626                     MON:   EQU  0E626H
B900                     START: EQU  0B900H
                         ;
                                ORG  START
                         ;
B900 DB09                MAIN:  IN   A,(09H);I/O ﾎﾟｰﾄ ﾉ 09H ｶﾗ ﾃﾞｰﾀ ｦ ﾖﾐﾄﾙ
B902 FEBF                       CP   0BFH   ;ｽﾍﾟｰｽ ｷｰ ｶﾞｵｻﾚﾀﾃﾞｰﾀﾄ ｲｯﾁ ｽﾙｶﾋｶｸｽﾙ
B904 CA26E6                     JP   Z,MON  ; ｽﾍﾟｰｽ ｷｰ ｶﾞ ｵｻﾚﾃｲﾚﾊﾞ ﾓﾆﾀｰ ﾍ ﾓﾄﾞﾙ
                         ;
B907 0E05                       LD   C,05H  ;ｼﾞｶﾝ ﾏﾁ ﾉ ﾙｰﾁﾝ
B909 06FF                LOP2:  LD   B,0FFH
B90B 05                  LOP1:  DEC  B
B90C C20BB9                     JP   NZ,LOP1
B90F 0D                         DEC  C
B910 C209B9                     JP   NZ,LOP2
                         ;
B913 C300B9                     JP   MAIN
                         ;
B916                            END
```

*Section 3*

# 論理演算命令を使ってプログラムを作る

Keyboard

## 論理演算命令は，大きく3種類

**リスト3-1**で分岐条件を作っているのが，CP(コンペア)命令です．この命令は非常に便利で，大きい，等しい，小さい，が区別出来るものでした．今度は，CP 命令の代りに「論理演算命令」と呼ばれる命令を使用してみることにしましょう．

まず，論理演算命令について説明しましょう．

論理演算命令は，1ビットごとの論理演算を行うもので，マシン語では大切な命令のひとつです．論理演算命令は，大きく分けて次の3命令に分けられます．

1．AND
2．OR
3．XOR

では，まず一番多く使用されるAND命令からみていきましょう．

## かけ算みたいなAND命令

論理演算だからといって，難しく考える必要はありません．数学の乗算と同じに考えて構いません．**表3-1**の計算をみてください．

**表3-1　AND命令の計算例**

| A | × | B | = | X |
|---|---|---|---|---|
| 0 | × | 0 | = | 0 |
| 1 | × | 0 | = | 0 |
| 0 | × | 1 | = | 0 |
| 1 | × | 1 | = | 1 |

このようになります．お互いに1のときだけ答は1になります．0があれば

答は0となります．2進法ですから，0と1以外の数字はありません．

具体的な例で説明しましょう．36HとFAHとのANDの論理演算をしてみましょう．

表3-2　36HとFAHのAND

| ビット数 | (36H) | | (FAH) | | 演算結果 |
|---|---|---|---|---|---|
| | A | × | B | = | X |
| 7 | 0 | × | 1 | = | 0 |
| 6 | 0 | × | 1 | = | 0 |
| 5 | 1 | × | 1 | = | 1 |
| 4 | 1 | × | 1 | = | 1 |
| 3 | 0 | × | 1 | = | 0 |
| 2 | 1 | × | 0 | = | 0 |
| 1 | 1 | × | 1 | = | 1 |
| 0 | 0 | × | 0 | = | 0 |

演算結果が32Hとなりましたが，大切なのは各1ビットです．

なおAND命令には，次の3種類があります．

## ■ 3つのAND命令

### AND r

書　式　AND　r（rはA，B，C，D，E，H，Lレジスタ）

機　能　Aレジスタとrで指定されたレジスタとで，各ビットごとにAND演算を行い，その結果をAレジスタにセットします．

例　AND　B

Aレジスタのデータを A3H，Bレジスタのデータを9BHとすると，

Aレジスタのデータ………1 0 1 0 0 0 1 1 B
Bレジスタのデータ………1 0 0 1 1 0 1 1 B
演算結果……………………1 0 0 0 0 0 1 1 B

となります．

フラグの変化　Sフラグ：命令を実行した結果，
最上位のビット（7ビット目）が1なら1
最上位のビット（7ビット目）が0なら0

となります.

Zフラグ：命令を実行した結果が

00Hのとき1

それ以外のとき0

となります.

P/Vフラグ：命令を実行した結果，1になっているビットの数が

偶数なら1

奇数なら0

となります.

Cフラグ：常に0となります.

## AND n

書　式　AND n　(n＝0～255)

機　能　Aレジスタとデータnとで各ビットごとにAND演算を行い，その結果をAレジスタにセットします.

例　AND 78H

Aレジスタのデータが A3H，データnを78Hとすると，

Aレジスタのデータ………1 0 1 0 0 0 1 1 B

データn……………………0 1 1 1 1 0 0 0 B

演算結果……………………0 0 1 0 0 0 0 0 B

となります.

フラグの変化　Sフラグ：命令を実行した結果，

最上位のビット（7ビット目）が1なら1

最上位のビット（7ビット目）が0なら0

となります.

Zフラグ：命令を実行した結果が

00Hのとき1

それ以外のとき0

となります.

P/Vフラグ：命令を実行した結果，1になっているビットの数が

偶数なら1

奇数なら0

となります．

Cフラグ：常に0となります．

---

AND　(HL)

書　式　AND　(HL)

機　能　Aレジスタと(HL)で指定された番地のデータとで，各ビットごとにAND演算を行い，その結果をAレジスタにセットします．

例　AND　(HL)

Aレジスタのデータが A3H, HL ペアレジスタが指定する番地のデータを3AHとすると，

```
Aレジスタのデータ………1 0 1 0 0 0 1 1 B
（HL）のデータ…………0 0 1 1 1 0 1 0 B
─────────────────────────────
演算結果………………………0 0 1 0 0 0 1 0 B
```

となります．

フラグの変化　Sフラグ：命令を実行した結果，

最上位のビット（7ビット目）が1なら1

最上位のビット（7ビット目）が0なら0

となります．

Zフラグ：命令を実行した結果が

00Hのとき1

それ以外のとき0

となります．

P/Vフラグ：命令を実行した結果，1になっているビットの数が

偶数なら1

奇数なら0

となります．

Cフラグ：常に0となります．

## たし算みたいなOR命令

このOR命令は，数学の加算と同じに考えてかまいません．**表3-3**の計算式を

みてください．

表3-3　OR命令の計算例

| A | + | B | = | X |
|---|---|---|---|---|
| 0 | + | 0 | = | 0 |
| 0 | + | 1 | = | 1 |
| 1 | + | 0 | = | 1 |
| 1 | + | 1 | = | 1 |

このようになります．どちらかが1であれば，答は1になります．この場合は，両方とも0のときだけ答が0になります．具体的な例で説明しましょう．

36HとFAHとのORの論理演算をしてみましょう．

表3-4　36HとFAHのOR

| ビット数 | (36H) | | (FAH) | | 演算結果 |
|---|---|---|---|---|---|
| | A | + | B | = | X |
| 7 | 0 | + | 1 | = | 1 |
| 6 | 0 | + | 1 | = | 1 |
| 5 | 1 | + | 1 | = | 1 |
| 4 | 1 | + | 1 | = | 1 |
| 3 | 0 | + | 1 | = | 1 |
| 2 | 1 | + | 0 | = | 1 |
| 1 | 1 | + | 1 | = | 1 |
| 0 | 0 | + | 0 | = | 0 |

演算結果がFEHとなりました．この命令は，任意のビットを1にしたいときに多く使用されます．

## ■ 3つのOR命令

なお，OR命令には，次の3種類があります．

**OR r**

書　式　　OR　r（rはA，B，C，D，E，H，Lレジスタ）

機　能　　Aレジスタとrで指定されたレジスタとで，各ビットごとにOR演算を行い，その結果をAレジスタにセットします．

例　　OR　B

Aレジスタのデータが A3H，Bレジスタのデータが 9BH とすると

Aレジスタのデータ……… 1 0 1 0 0 0 1 1 B
Bレジスタのデータ……… 1 0 0 1 1 0 1 1 B
演算結果……………………… 1 0 1 1 1 0 1 1 B

となります．桁上がりは無視されます．

フラグの変化　Sフラグ：命令を実行した結果が
最上位のビット（7ビット目）が1なら1
最上位のビット（7ビット目）が0なら0
となります．

Zフラグ：命令を実行した結果が
00Hのとき1
それ以外のとき0
となります．

P/Vフラグ：命令を実行した結果，1になっているビットの数が
偶数なら1
奇数なら0
となります．

Cフラグ：常に0となります．

## OR n

書　式　OR　n　(n=0〜255)

機　能　Aレジスタとデータnとで各ビットごとにOR演算を行い，その結果をAレジスタにセットします．

例　OR　78H

Aレジスタのデータが A3H，データnを78Hとすると，

Aレジスタのデータ……… 1 0 1 0 0 0 1 1 B
データn……………………… 0 1 1 1 1 0 0 0 B
演算結果……………………… 1 1 1 1 1 0 1 1 B

となります．

フラグの変化　Sフラグ：命令を実行した結果が
最上位のビット（7ビット目）が1なら1

最上位のビット（7ビット目）が0なら0
となります.
Zフラグ：命令を実行した結果が
00Hのとき1
それ以外のとき0
となります.
P/Vフラグ：命令を実行した結果，1になっているビットの数が
偶数なら1
奇数なら0
となります.
Cフラグ：常に0となります.

## OR (HL)

書　式　OR (HL)

機　能　Aレジスタと(HL)で指定された番地のデータとで各ビットごとにOR演算を行い，その結果をAレジスタにセットします.

OR (HL)

Aレジスタが A3H, HLペアレジスタが指定する番地のデータを3AHとすると，

| | |
|---|---|
| Aレジスタのデータ……… | 1 0 1 0 0 0 1 1 B |
| （HL）のデータ………… | 0 0 1 1 1 0 1 0 B |
| 演算結果……………………… | 1 0 1 1 1 0 1 1 B |

となります.

フラグの変化　Sフラグ：命令を実行した結果，
最上位のビット（7ビット目）が1なら1
最上位のビット（7ビット目）が0なら0
となります.
Zフラグ：命令を実行した結果が
00Hのとき1
それ以外のとき0
となります.
P/Vフラグ：命令を実行した結果，1になっているビットの数が

偶数なら1
奇数なら0
となります．
Cフラグ：常に0となります．

## 等しければ0，XOR命令

このXOR命令は，数学でたとえるなら何でたとえれば良いでしょうか．ちょっと思い浮かびません．この命令は，対応する2つのビットが共に等しければ0となり，違っていれば1となるものです．**表3-5**の計算式を見てください．

**表3-5　XOR命令の計算例**

| A | ? | B | = | X |
|---|---|---|---|---|
| 0 | ? | 0 | = | 0 |
| 0 | ? | 1 | = | 1 |
| 1 | ? | 0 | = | 1 |
| 1 | ? | 1 | = | 0 |

このように，両方のビットが等しければ答は0になり，違っていれば答は1になります．具体的な例で説明しましょう．36HとFAHとのXORの論理演算をしてみましょう．

演算結果がCCHとなりました．この命令は，任意のビットを反転したりするときなどに使われます．

**表3-6　36HとFAHのXOR**

| ビット数 | (36H) | | (FAH) | | 演算結果 |
|---|---|---|---|---|---|
| | A | ? | B | = | X |
| 7 | 0 | ? | 1 | = | 1 |
| 6 | 0 | ? | 1 | = | 1 |
| 5 | 1 | ? | 1 | = | 0 |
| 4 | 1 | ? | 1 | = | 0 |
| 3 | 0 | ? | 1 | = | 1 |
| 2 | 1 | ? | 0 | = | 1 |
| 1 | 1 | ? | 1 | = | 0 |
| 0 | 0 | ? | 0 | = | 0 |

## ■ 3つのXOR命令

XOR命令には，次の3種類があります．

### XOR r

書　式　XOR　r（rはA，B，C，D，E，H，Lレジスタ）

機　能　Aレジスタとrで指定されたレジスタとで，各ビットごとにXOR演算を行い，その結果をAレジスタにセットします．

例　XOR　B

Aレジスタのデータ………1 0 1 0 0 0 1 1 B
Bレジスタのデータ………1 0 0 1 1 0 1 1 B
演算結果……………………0 0 1 1 1 0 0 0 B

となります．

フラグの変化

Sフラグ：命令を実行した結果が
最上位のビット（7ビット目）が1なら1
最上位のビット（7ビット目）が0なら0
となります．

Zフラグ：命令を実行した結果が
00Hのとき1
それ以外のとき0
となります．

P/Vフラグ：命令を実行した結果，1になっているビットの数が
偶数なら1
奇数なら0
となります．

Cフラグ：常に0となります．

### XOR n

書　式　XOR　n　（n=0～255）

機　能　Aレジスタとnデータとで各ビットごとにXOR演算を行い，その結果をAレジスタにセットします．

例　XOR　78H

Ａレジスタのデータに A3H がセットされているとし，ｎのデータが 78H とすると，

Ａレジスタのデータ………１０１０００１１Ｂ
ｎのデータ…………………０１１１１０００Ｂ
演算結果……………………１１０１１０１１Ｂ

となります．

フラグの変化　Ｓフラグ：命令を実行した結果が
最上位のビット（７ビット目）が１なら１
最上位のビット（７ビット目）が０なら０
となります．

Ｚフラグ：命令を実行した結果が
00Ｈのとき１
それ以外のとき０
となります．

P/V フラグ：命令を実行した結果，１になっているビットの数が
偶数なら１
奇数なら０
となります．

Ｃフラグ：常に０となります．

## XOR　(HL)

書　式　XOR（HL）

機　能　ＡレジスタとＨＬペアレジスタが指定する番地のデータとで各ビットごとに XOR 演算を行い，その結果をＡレジスタにセットします．

例　XOR（HL）

Ａレジスタのデータに Ａ３Ｈがセットされているとし，HL ペアレジスタが指定する番地のデータが A3H とすると

Ａレジスタのデータ………１０１０００１１Ｂ
（ＨＬ）のデータ…………００１１１０１０Ｂ
演算結果……………………１００１１００１Ｂ

となります．

| | |
|---|---|
| フラグの変化 | Sフラグ：命令を実行した結果が<br>　　最上位のビット（7ビット目）が1なら1<br>　　最上位のビット（7ビット目）が0なら0<br>となります．<br>Zフラグ：命令を実行した結果が<br>　　00Hのとき1<br>　　それ以外のとき0<br>となります．<br>P/Vフラグ：命令を実行した結果，1になっているビットの数が<br>　　偶数なら1<br>　　奇数なら0<br>となります．<br>Cフラグ：常に0となります． |

## 1ビットごとの論理演算でドット単位の移動が可能に

論理演算の特徴は何と言っても，ビット単位の演算でしょう．つまり，1ビット単位の情報を扱うのです．使用範囲は結構広く，たとえばグラフィックスに使います．PC-8801mkIISRのグラフィックス画面では，VRAMは1ビットが1ドットに対応しています．したがって，1ビットごとの論理演算を用いれば，ドット単位で画面を動かすことも可能になります．論理命令は是非とも覚えなければならない命令なのです．

### ■任意のビットを取り出したり消したり～AND

この命令は，特定のビットを取り出したいときや，特定のビットを消したいときに使用されます．もし，任意のビットだけを取り出したいのなら，取り出したいビットを1にして，この命令を実行すれば良いのです．1にしたビットだけを取り出せます．具体例を図3-5に示します．

図3-5　ANDを使って任意のビットをとり出す

AND演算を行うと

演算の結果…………………　0　0　1　0　0　0　0　0

この場合は取り出すデータの5ビット目が1なので，演算の結果5ビット目が1になった．もし，取り出すデータの5ビット目が0ならば，演算の結果5ビット目は0となる．

このAND命令によるビットの取り出しは，他のビットに関係していませんので，この方法とは反対に，任意のビットだけを消すこともできます．つまり，消したいビットを0にして他のビットは1にします．すると，0にしたビットだけを消すことができます．

### ■任意のビットを1にする～OR

この命令の目的は，任意のビットを1にすることです．この点はAND命令とは逆です．AND命令のときのようにセットしたいビットを1にして，OR命令を実行すれば良いのです．このOR命令では，任意のデータを消すことはできません．この命令の実行のやり方はAND命令と同じです．

### ■"XOR 0FFH"でもとのデータを反転

この命令は比較的使用されることの少ない命令です．"XOR 0FFH"で使用すると，もとのデータが反転されます．

## 論理演算の特殊な計算

論理演算の特殊な計算を説明しましょう．それは次の3命令で，最も多く出てくるものです．

```
AND  A
OR   A
XOR  A
```

つまりAレジスタ同士の論理演算です.「レジスタ同士で論理演算をやって何になるのだろう」と思う人もいるかもしれませんが，ちゃんとある目的を果たしてくれるのです．

### ■Cyフラグをリセットする"AND A"，"OR A"

まず，"AND A"，"OR A"を実行してみます（図3-6）．

図3-6 AND A，OR Aの具体例

AND A

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ビット |
|---|---|---|---|---|---|---|---|---|---|
| Aレジスタのデータ……… | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | |
| Aレジスタのデータ……… | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | |
| 演算の結果…………………… | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | |

結果は元のデータと同じ．

どちらの命令の結果も，もとのデータと同じになってしまいました．結果は同じでも，しっかり計算をやってくれるのです．

何をするかと言うと，Cyフラグのリセットです．この命令を実行するとCyフラグをリセットしてくれるのです．つまり，

> Cyフラグをリセットするために，“AND A”，“OR A”を使用する

となります．

## ■データが00Hかどうか調べるときにも使える

もうひとつは，Zフラグの利用です．たとえば，あるメモリのデータが00Hであるかどうかを調べるときに“AND A”，“OR A”は利用されます．

まず，LD命令などで調べたいメモリのデータをAレジスタにセットします．その後で，“AND A”，“OR A”のどちらかの命令を実行します．もし，実行結果が0であれば，Zフラグは1となります．00H以外のデータであれば，Zフラグは0となります．

## ■“XOR　A”でAレジスタをクリア

次に“XOR A”命令を見てみましょう．図3-7を見てください．

図3-7　XOR　Aの具体例

XOR　A

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ビット |
|---|---|---|---|---|---|---|---|---|---|
| Aレジスタのデータ……… | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | |
| Aレジスタのデータ……… | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | |
| 演算の結果…………………… | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

結果はすべて0になってしまう．

このように，すべてのビットは0になります．つまり，

| Aレジスタをクリア（00H）するために，"XOR A"を使用する |
|---|

となります．Aレジスタをクリアする方法は"LD A, 00H"などがありますが，"XOR A"は1バイトで良いので，1バイト分得します．ただし，この命令はZフラグやCyフラグをリセットしてしまうので，注意が必要です．

## キー入力のテストプログラムを作る

今まで長々と論理演算命令を説明してきましたが，グラフィックスの章に入る前に練習をしておこうと考えたからです．

話をもとに戻して，論理演算命令を使用してリスト3-1を作ってみましょう．CP命令を使用していたのを，AND命令に変えるだけです．

スペース・キーが押されたときのデータは図3-8のようになります．

**図3-8 スペース・キーが押されたときのデータ**

スペース・キーが押されたときのデータは
I/Oポート・アドレス……………………………… 09H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ビット |
|---|---|---|---|---|---|---|---|---|---|
| データ………………… | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | (BFH) |

AND命令でZフラグが1になることを利用します．つまり，全部のビットのデータを0にするのです．全部のビットを消すには，6ビット目を1にして他のビットを0にします（図3-9）．

**図3-9 全部のビットを消す**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ビット |
|---|---|---|---|---|---|---|---|---|---|
| スペース・キーのデータ……… | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | (BFH) |
| AND命令で用意するデータ…… | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | (40H) |
| 演算の結果……………………… | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

プログラムはリスト3-2のようになります。

**●リスト3-2　論理演算を使ったキー入力テスト**

```
                    ;
                    ;********* KEY IN TEST **************
                    ;
                    ;PROGRAM NAME --->keyt2.e
                    ;
                    ;ｷｰ ﾆｭｳﾘｮｸ ﾉ ﾃｽﾄ ﾌﾟﾛｸﾞﾗﾑ ( AND 40H )
                    ;
                    ;***********************************
                    ;
E626                MON:   EQU  0E626H
B900                START: EQU  0B900H
                    ;
                           ORG  START
                    ;
B900 CD8B42                CALL 428BH
B903 DB09           MAIN:  IN   A,(09H);I/O ﾎﾟｰﾄ ﾉ 09H ｶﾗ ﾃﾞｰﾀ ｦ ﾖﾐﾄﾙ
B905 E640                  AND  40H    ;ｽﾍﾟｰｽ ｷｰｶﾞｵｻﾚﾀﾃﾞｰﾀﾄ ｲｯﾁ ｽﾙｶ ﾋｶｸ ｽ
B907 CA26E6                JP   Z,MON  ; ｽﾍﾟｰｽ ｷｰ ｶﾞ ｵｻﾚﾃｲﾚﾊﾞ ﾓﾆﾀｰ ﾍ ﾓﾄﾞﾙ
                    ;
B90A 0E02                  LD   C,02H  ;ｼﾞｶﾝ ﾏﾁ ﾉ ﾙｰﾁﾝ
B90C 06FF           LOP2:  LD   B,0FFH
B90E 05             LOP1:  DEC  B
B90F C20EB9                JP   NZ,LOP1
B912 0D                    DEC  C
B913 C20CB9                JP   NZ,LOP2
                    ;
B916 C303B9                JP   MAIN
                    ;
B919                       END
```

*Section 4*

# キー入力とキャラクタの移動プログラムを作る

Keyboad

## キー入力とキャラクタの4方向移動プログラムを作る

キーボードからの入力原理が理解できたところで，実際のゲームプログラムなどで使用されている，キー入力とキャラクタの移動プログラムを作ってみましょう．

比較的良く使用されているのが，4方向の移動ですね．この原理さえ理解できれば，キー入力で頭を悩ますことがなくなります．

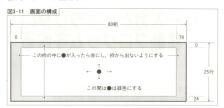
### ■移動に使う 4 つのキー

この4方向のキー入力でよく利用されるキーは，右側にあるテンキー（数値入力用キー）です．4方向のキー入力には，**図3-10**のようなキーが利用されています．

**図3-10　使用する4つのキー**

### ■画面の構成

次の画面の構成はプログラムを簡単にするために，**図3-11**のようなものとします．画面の隅にきたら●を画面からはみ出さないようにしなければなりません．そこで，警告のために，画面の隅にきたら●を赤に変えます．

図3-11　画面の構成

## ■インデックス・レジスタを使う

汎用レジスタの中でポインタ(メモリなどを間接的，直接的に指定する役割)として使用できるのは，HL ペアレジスタしかありません．しかし，Z-80ではこれを補うためにポインタ専用のインデックス・レジスタ（IX，IY）を持っているのです．このレジスタのおかげで，アドレッシングが非常に楽になります．

インデックス・レジスタ（IX，IY）の使い方は HL ペアレジスタとほとんど同じですが，IX，IY は16ビットですので，8ビットずつ分けて使うことはできません．また，残念なことに16ビットの減算命令がありません．

インデックス・レジスタ（IX，IY）は，HL ペアレジスタにない便利なアドレッシングを持っています．レジスタのデータの値を変えずに±128の範囲で，アドレッシングをすることができるのです．つまり，オフセットしたことと同じになります．

```
LD  (IX+d), A
         ↑
         これはオフセット値を示します．
```

8ビットの符号付2進数を示しましょう．

8ビットの符号付2進数の例

＋方向

| 0 | 1 | 2 | 3 | ………………………… | 125 | 126 | 127 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | ………………………… | 7D | 7E | 7F |

ー方向

| −1 | −2 | −3 | −4 | ………………………… | −126 | −127 | −128 |
|---|---|---|---|---|---|---|---|
| FF | FE | FD | FC | ………………………… | 82 | 81 | 80 |

例.

LD　IX, 0F418H　←　インデックス・レジスタのIXにF418Hをセットします.

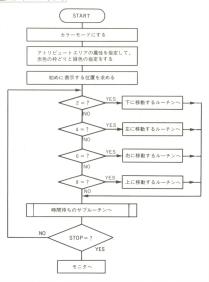
LD　(IX＋1), A　←　F418H＋1HしたアドレスのF419HにAレジスタのデータをセットします.

LD　(IX＋FEH), A　←　F418H－02HしたアドレスのF416HにAレジスタのデータをセットします.

このアドレッシングは，レジスタのデータを変えないので，VRAMのアドレス指定に便利です．この節のサンプルプログラムでも，アトリビュートエリアの属性の指定に使用しています．IYインデックス・レジスタは，IXインデックス・レジスタと全く同じです．

## ■プログラムのフローチャートを作る

ラフ・フローチャートは図3-12のようになります．まず，各レジスタの用途は次の通りです．

図3-12　ラフ・フローチャート

HL ペアレジスタ……………………VRAM のポインタ
IX インデックス・レジスタ………アトリビュートエリアの属性エリアのポインタ
D レジスタ…………………………キャラクタの横の桁のカウンタ
E レジスタ…………………………キャラクタの縦の行のカウンタ
A レジスタ…………………………キャラクタの表示または消去

## ■ HL ペアレジスタのデータとキャラクタの移動の関係

キャラクタのアドレッシングは HL ペアレジスタで行っており，キャラクタの移動と HL ペアレジスタとの関係は**図3-13**のようになります．

**図3-13 キャラクタ移動とHLペアレジスタ**

また，D，E レジスタは移動した数を数えており，画面からはみ出さないようにしています．

## ■アトリビュートの属性

アトリビュートの属性は，**図3-14**のようにしなくてはなりません．

**図3-14 アトリビュートの属性**

このように，2行目から23行目までは属性を3回指定しなければなりません．1行について6バイトのデータを書き込むことになりますが，この際，IX インデックス・レジスタが便利なのです．

## ■ Cy フラグのリセット　SCF，CCF 命令

Cy フラグをリセットするのに“AND A”命令が使用できますが，別の命令を使用してみましょう．次の2命令を組み合わせて使用します．

| |
|---|
| SCF………Cy フラグを強制的に1にする |
| CCF………Cy フラグを反転する |

SCF, CCFの2命令を用いてCyフラグをリセットすれば，他のフラグはそのままになっています．したがって，他のフラグを壊したくないときに使用します．また，数値演算のときにも使われています．

## ■キャラクタ4方向移動プログラム

特に難しくはないので，よく理解してください．なお,**リスト3-4**のプログラムではこれを応用して，キャラクタを8方向に移動させています．

**●リスト3-3　キー入力とキャラクタ4方向移動プログラム**

```
                    ;***** COLOR TEXT MODE *****
                    ;
                    ;PROGRAM NAME ----> keyin.e
                    ;
                    ; ｷｰﾆｭｳﾘｮｸ ﾃﾞﾓ ﾌﾟｸﾞﾗﾑ 1(4 ﾎｳｺｳ)
                    ;
                    ;
                    ;.......................................
                    ;
                    ;ｷｬﾗｸﾀｰ ﾉ ｲﾄﾞｳ ｷｰ
                    ;
                    ;                 8
                    ;                 |
                    ;             4---●---6
                    ;                 |
                    ;                 2
                    ;                      STOP ｷｰ ﾃﾞ ﾓﾆﾀｰ ﾍ
                    ;........................................
                    ;
                    ;
B900                START: EQU  0B900H          ;PROGRAM START ADDRESS
E626                MON:   EQU  0E626H          ;ﾓﾆﾀｰ ﾉ ｴﾝﾄﾘ ﾎﾟｲﾝﾄ
6F6B                COLOR: EQU  6F6BH           ;CRT ｺﾝﾄﾛｰﾙ ﾙｰﾁﾝ
00EC                CHR:   EQU  0ECH            ; ｷｬﾗｸﾀｰ ﾉ ｺｰﾄﾞ
                    ;
                    ;
                    ;
                    ;
                           ORG  START
                    ;
                    ;CRT COLOR MODE ------------------------------------
                    ;
```

```
B900 011950   START: LD   BC,5019H
B903 3E01            LD   A,01H
B905 32B2E6          LD   (0E6B2H),A
B908 3E19            LD   A,19H
B90A 32B3E6          LD   (0E6B3H),A
B90D 3EE8            LD   A,0E8H
B90F 32B4E6          LD   (0E6B4H),A
B912 3E00            LD   A,00H
B914 32B8E6          LD   (0E6B8H),A
B917 3EFF            LD   A,0FFH
B919 32B9E6          LD   (0E6B9H),A
B91C CD6B6F          CALL COLOR
                ;
                ; ﾜｸ ﾄﾞﾘ ｦ ｱｶ ﾆ ｽﾙ ------------------------
                ;
B91F DD2118F4        LD   IX,0F418H
B923 DD360000        LD   (IX+0),00H          ;ｳｴ ﾉ ｶﾗｰ
B927 DD360148        LD   (IX+1),48H          ;ｱｶ
B92B 0617            LD   B,17H
B92D 117800          LD   DE,0078H
B930 DD19     LOPI:  ADD  IX,DE
B932 DD360000        LD   (IX+0),00H          ;ﾋﾀﾞﾘ ﾊｼ ﾉ ｶﾗｰ
B936 DD360148        LD   (IX+1),48H          ;ｱｶ
B93A DD360201        LD   (IX+2),01H          ;ﾁｭｳｶﾝ ﾉ ｶﾗｰ
B93E DD360388        LD   (IX+3),088H         ;ｷｬﾗｸﾀｰ ﾉ ｶﾗｰ  ﾐﾄﾞﾘ
B942 DD36044F        LD   (IX+4),4FH          ;ﾐｷﾞﾊｼ ﾉ ｶﾗｰ
B946 DD360548        LD   (IX+5),48H          ;ｱｶ
B94A 05              DEC  B
B94B C230B9          JP   NZ,LOPI
B94E DD19            ADD  IX,DE
B950 DD360000        LD   (IX+0),00H          ;ｼﾀ ﾉ ｶﾗｰ
B954 DD360148        LD   (IX+1),48H          ;ｱｶ
                ;
                ;ｶｰｿﾙ ｦ ｹｽ ﾙｰﾁﾝ ｦ ﾖﾌﾞ --------------------------------
                ;
B958 CD8B42          CALL 428BH
                ;
                ;
                ; MAIN --------------------------------------------
                ;
B95B 110000          LD   DE,0000H ;Dﾚｼﾞｽﾀ ﾖｺ ﾉ ｹﾀ ,Eﾚｼﾞｽﾀ ﾀﾃ ﾉ ｷﾞｮｳ
B95E 21C8F3          LD   HL,0F3C8H;CRT ﾉ ﾋﾀﾞﾘ ｳｴ ﾉ ｲﾁ
B961 3E27            LD   A,27H
B963 23       LOPM:  INC  HL
B964 14              INC  D
B965 3D              DEC  A
B966 C263B9          JP   NZ,LOPM
B969 3E0D            LD   A,0DH
B96B 017800          LD   BC,0078H
B96E 09       LOPN:  ADD  HL,BC
B96F 1C              INC  E
B970 3D              DEC  A
B971 C26EB9          JP   NZ,LOPN
                ;
B974 3EEC            LD   A,CHR      ; ｷｬﾗｸﾀｰ ｦ ﾏﾝﾅｶ ﾆ ﾋｮｳｼﾞ ｽﾙ
B976 77              LD   (HL),A
                ;
B977 DB00     LOOPK: IN   A,(00H)    ;2 ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B979 47              LD   B,A
B97A E604            AND  04H
B97C CA9FB9          JP   Z,DOWN
```

```
                ;
B97F 78                 LD    A,B       ;4 ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B980 E610               AND   10H
B982 CAB2B9             JP    Z,LEFT
                ;
B985 78                 LD    A,B       ;6 ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B986 E640               AND   40H
B988 CAC2B9             JP    Z,RIGHT
                ;
B98B DB01               IN    A,(01H)   ;8 ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B98D E601               AND   01H
B98F CAD2B9             JP    Z,UP
                ;
B992 CDE8B9     LOPE:   CALL  WAIT
                ;
B995 DB09               IN    A,(09H)   ;STOP ｷｰ ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B997 E601               AND   01H
B999 C277B9             JP    NZ,LOOPK
                ;
B99C C326E6             JP    MON       ; ﾓﾆﾀｰ ﾍ ﾓﾄﾞﾙ
                ;
                ; ● ｲﾄﾞｳ ﾙｰﾁﾝ -----------------------------------
                ;
B99F 7B         DOWN:   LD    A,E       ;ｼﾀ ﾍ ｲﾄﾞｳ ｽﾙ ﾙｰﾁﾝ
B9A0 FE18               CP    18H
B9A2 CA92B9             JP    Z,LOPE
B9A5 AF                 XOR   A         ;A ﾚｼﾞｽﾀ ｦ ｸﾘｱ ｽﾙ
B9A6 77                 LD    (HL),A
B9A7 017800             LD    BC,0078H
B9AA 09                 ADD   HL,BC
B9AB 3EEC               LD    A,CHR
B9AD 77                 LD    (HL),A
B9AE 1C                 INC   E
B9AF C392B9             JP    LOPE
                ;
B9B2 7A         LEFT:   LD    A,D                 ;ﾋﾀﾞﾘ ﾍ ｲﾄﾞｳ ｽﾙ ﾙｰﾁﾝ
B9B3 FE00               CP    00H
B9B5 CA92B9             JP    Z,LOPE
B9B8 AF                 XOR   A
B9B9 77                 LD    (HL),A
B9BA 2B                 DEC   HL
B9BB 3EEC               LD    A,CHR
B9BD 77                 LD    (HL),A
B9BE 15                 DEC   D
B9BF C392B9             JP    LOPE
                ;
B9C2 7A         RIGHT:  LD    A,D                 ;ﾐｷﾞ ﾍ ｲﾄﾞｳ ｽﾙ ﾙｰﾁﾝ
B9C3 FE4F               CP    4FH
B9C5 CA92B9             JP    Z,LOPE
B9C8 AF                 XOR   A
B9C9 77                 LD    (HL),A
B9CA 23                 INC   HL
B9CB 3EEC               LD    A,CHR
B9CD 77                 LD    (HL),A
B9CE 14                 INC   D
B9CF C392B9             JP    LOPE
                ;
B9D2 7B         UP:     LD    A,E                 ;ｳｴ ﾍ ｲﾄﾞｳ ｽﾙ ﾙｰﾁﾝ
B9D3 FE00               CP    00H
B9D5 CA92B9             JP    Z,LOPE
B9D8 AF                 XOR   A
```

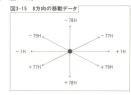
```
B9D9 77                  LD    (HL),A
B9DA 017800              LD    BC,0078H
B9DD 37                  SCF                          ;ｷｬﾘｰﾌﾗｸﾞ ｦ 1 ﾆｽﾙ
B9DE 3F                  CCF                          ;ｷｬﾘｰﾌﾗｸﾞ ｦ 0 ﾆｽﾙ
B9DF ED42                SBC   HL,BC                  ;(ｷｬﾘｰﾌﾗｸﾞ ﾓ ﾋｸ )
B9E1 3EEC                LD    A,CHR
B9E3 77                  LD    (HL),A
B9E4 1D                  DEC   E
B9E5 C392B9              JP    LOPE
                ;
                ;ｼﾞｶﾝ ﾏﾁ ﾉ ﾙｰﾁﾝ -------------------------------------
                ;
B9E8 0E0F       WAIT:    LD    C,0FH
B9EA 06FF       WLP2:    LD    B,0FFH
B9EC 05         WLP1:    DEC   B
B9ED C2ECB9              JP    NZ,WLP1
B9F0 0D                  DEC   C
B9F1 C2EAB9              JP    NZ,WLP2
B9F4 C9                  RET
                ;
B9F5                     END
```

## 8方向の移動プログラムを作る

前のプログラムを少し複雑にして，8方向に移動するプログラムにしてみましょう．プログラムはほとんどそのままで，斜めに移動するルーチンを追加しただけです．

### ■斜めに移動するには

斜めに移動する方法は簡単で，HLペアレジスタに加減するデータ78Hを1プラス，マイナスするだけです（図3-15）．

図3-15　8方向の移動データ

## ■使用するキー

使用するキーは，1，2，3，4，6，7，8，9のキーです．5のキーは使用しません（図3-16）．

図3-16　使用するキー

●リスト3-4　キー入力とキャラクタ8方向移動プログラム

```
                    ;***** COLOR TEXT MODE *****
                    ;
                    ;PROGRAM NAME ----> keyin8.e
                    ;
                    ; ｷｰﾆｭｳﾘｮｸ ﾃﾞﾓ ﾌﾟｸﾞﾗﾑ 2(8 ﾎｳｺｳ)
                    ;
                    ;........................................
                    ;
                    ;ｷｬﾗｸﾀｰ ﾉ ｲﾄﾞｳ ｷｰ
                    ;
                    ;               7  8  9
                    ;                \ | /
                    ;               4--●--6
                    ;                / | \
                    ;               1  2  3
                    ;                       STOP ｷｰ ﾃﾞ ﾓﾆﾀｰ ﾍ
                    ;........................................
                    ;
                    ;
B900                START:  EQU   0B900H          ;PROGRAM STRAT ADDRESS
E626                MON:    EQU   0E626H          ;ﾓﾆﾀｰ ﾉ ｴﾝﾄﾘ ﾎﾟｲﾝﾄ
6F6B                COLOR:  EQU   6F6BH           ;CRT ｺﾝﾄﾛｰﾙ ﾙｰﾁﾝ
00EC                CHR:    EQU   0ECH            ; ｷｬﾗｸﾀｰ ﾉ ｺｰﾄﾞ
                    ;
                    ;
                    ;
                    ;
                            ORG   START
                    ;
                    ;CRT COLOR MODE -----------------------------------
                    ;
```

```
B900 011950   START: LD   BC,5019H
B903 3E01            LD   A,01H
B905 32B2E6          LD   (0E6B2H),A
B908 3E19            LD   A,19H
B90A 32B3E6          LD   (0E6B3H),A
B90D 3EE8            LD   A,0E8H
B90F 32B4E6          LD   (0E6B4H),A
B912 3E00            LD   A,00H
B914 32B8E6          LD   (0E6B8H),A
B917 3EFF            LD   A,0FFH
B919 32B9E6          LD   (0E6B9H),A
B91C CD6B6F          CALL COLOR
              ;
              ; ﾜｸ ﾄﾞﾘ ｦ ｱｶ ﾆ ｽﾙ ------------------------
              ;
B91F DD2118F4        LD   IX,0F418H
B923 DD360000        LD   (IX+0),00H            ;ｳｴ ﾉ ｶﾗｰ
B927 DD360148        LD   (IX+1),48H            ;ｱｶ
B92B 0617            LD   B,17H
B92D 117800          LD   DE,0078H
B930 DD19     LOPI:  ADD  IX,DE
B932 DD360000        LD   (IX+0),00H            ;ﾋﾀﾞﾘ ﾊｼ ﾉ ｶﾗｰ
B936 DD360148        LD   (IX+1),48H            ;ｱｶ
B93A DD360201        LD   (IX+2),01H            ;ﾁｭｳｶﾝ ﾉ ｶﾗｰ
B93E DD360388        LD   (IX+3),088H           ;ｷｬﾗｸﾀｰ ﾉ ｶﾗｰ  ﾐﾄﾞﾘ
B942 DD36044F        LD   (IX+4),4FH            ;ﾐｷﾞﾊｼ ﾉ ｶﾗｰ
B946 DD360548        LD   (IX+5),48H            ;ｱｶ
B94A 05              DEC  B
B94B C230B9          JP   NZ,LOPI
B94E DD19            ADD  IX,DE
B950 DD360000        LD   (IX+0),00H            ;ｼﾀ ﾉ ｶﾗｰ
B954 DD360148        LD   (IX+1),48H            ;ｱｶ
              ;
              ;ｶｰｿﾙ ｦ ｹｽ ﾙｰﾁﾝ ｦ ﾖﾌﾞ --------------------------------
              ;
B958 CD8B42          CALL 428BH
              ;
              ;
              ; MAIN ---------------------------------------------
              ;
B95B 110000          LD   DE,0000H  ;Dﾚｼﾞｽﾀ ﾖｺ ﾉ ｹﾀ ,Eﾚｼﾞｽﾀ ﾀﾃ ﾉ ｷﾞｮｳ
B95E 21C8F3          LD   HL,0F3C8H ;CRT ﾉ ﾋﾀﾞﾘ ｳｴ ﾉ ｲﾁ
B961 3E27            LD   A,27H
B963 23       LOPM:  INC  HL
B964 14              INC  D
B965 3D              DEC  A
B966 C263B9          JP   NZ,LOPM
B969 3E0D            LD   A,0DH
B96B 017800          LD   BC,0078H
B96E 09       LOPN:  ADD  HL,BC
B96F 1C              INC  E
B970 3D              DEC  A
B971 C26EB9          JP   NZ,LOPN
              ;
B974 3EEC            LD   A,CHR     ; ｷｬﾗｸﾀｰ ｦ ﾏﾝﾅｶ ﾆ ﾋｮｳｼﾞ ｽﾙ
B976 77              LD   (HL),A
              ;
B977 DB00     LOOPK: IN   A,(00H)   ;1 ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B979 47              LD   B,A
B97A E602            AND  02H
```

```
B97C CAB8B9          JP   Z,LDOWN
              ;
B97F 78              LD   A,B           ;2 ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B980 E604            AND  04H
B982 CAD2B9          JP   Z,DOWN
              ;
B985 78              LD   A,B           ;3 ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B986 E608            AND  08H
B988 CAE5B9          JP   Z,RDOWN
              ;
B98B 78              LD   A,B           ;4 ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B98C E610            AND  10H
B98E CAFFB9          JP   Z,LEFT
              ;
B991 78              LD   A,B           ;6 ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B992 E640            AND  40H
B994 CA0FBA          JP   Z,RIGHT
              ;
B997 78              LD   A,B           ;7 ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B998 E680            AND  80H
B99A CA1FBA          JP   Z,LUP
              ;
B99D DB01            IN   A,(01H)       ;8 ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B99F 47              LD   B,A
B9A0 E601            AND  01H
B9A2 CA3CBA          JP   Z,UP
              ;
B9A5 78              LD   A,B           ;9 ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B9A6 E602            AND  02H
B9A8 CA52BA          JP   Z,RUP
              ;
              ;
B9AB CD6FBA   LOPE:  CALL WAIT
              ;
B9AE DB09            IN   A,(09H)       ;STOP ｷｰ ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B9B0 E601            AND  01H
B9B2 C277B9          JP   NZ,LOOPK
              ;
B9B5 C326E6          JP   MON           ; ﾓﾆﾀｰ ﾍ ﾓﾄﾞﾙ
              ;
              ; ● ﾉ ｲﾄﾞｳ ﾙｰﾁﾝ -------------------------------------
              ;
B9B8 7B       LDOWN: LD   A,E           ;ﾋﾀﾞﾘｼﾀ ﾍ ｲﾄﾞｳ
B9B9 FE18            CP   18H
B9BB CAABB9          JP   Z,LOPE
B9BE 7A              LD   A,D
B9BF FE00            CP   00H
B9C1 CAABB9          JP   Z,LOPE
B9C4 AF              XOR  A
B9C5 77              LD   (HL),A
B9C6 017700          LD   BC,0077H
B9C9 09              ADD  HL,BC
B9CA 3EEC            LD   A,CHR
B9CC 77              LD   (HL),A
B9CD 1C              INC  E
B9CE 15              DEC  D
B9CF C3ABB9          JP   LOPE
              ;
B9D2 7B       DOWN:  LD   A,E           ;ｼﾀ ﾍ ｲﾄﾞｳ
B9D3 FE18            CP   18H
```

```
B9D5 CAABB9            JP    Z,LOPE
B9D8 AF                XOR   A
B9D9 77                LD    (HL),A
B9DA 017800            LD    BC,0078H
B9DD 09                ADD   HL,BC
B9DE 3EEC              LD    A,CHR
B9E0 77                LD    (HL),A
B9E1 1C                INC   E
B9E2 C3ABB9            JP    LOPE
                ;
B9E5 7B         RDOWN: LD    A,E              ;ミギシタ ヘ イドウ
B9E6 FE18              CP    18H
B9E8 CAABB9            JP    Z,LOPE
B9EB 7A                LD    A,D
B9EC FE4F              CP    4FH
B9EE CAABB9            JP    Z,LOPE
B9F1 AF                XOR   A
B9F2 77                LD    (HL),A
B9F3 017900            LD    BC,0079H
B9F6 09                ADD   HL,BC
B9F7 3EEC              LD    A,CHR
B9F9 77                LD    (HL),A
B9FA 1C                INC   E
B9FB 14                INC   D
B9FC C3ABB9            JP    LOPE
                ;
B9FF 7A         LEFT:  LD    A,D              ;ヒダリ ヘ イドウ
BA00 FE00              CP    00H
BA02 CAABB9            JP    Z,LOPE
BA05 AF                XOR   A
BA06 77                LD    (HL),A
BA07 2B                DEC   HL
BA08 3EEC              LD    A,CHR
BA0A 77                LD    (HL),A
BA0B 15                DEC   D
BA0C C3ABB9            JP    LOPE
                ;
BA0F 7A         RIGHT: LD    A,D              ;ミギ ヘ イドウ
BA10 FE4F              CP    4FH
BA12 CAABB9            JP    Z,LOPE
BA15 AF                XOR   A
BA16 77                LD    (HL),A
BA17 23                INC   HL
BA18 3EEC              LD    A,CHR
BA1A 77                LD    (HL),A
BA1B 14                INC   D
BA1C C3ABB9            JP    LOPE
                ;
BA1F 7B         LUP:   LD    A,E              ;ヒダリウエ ヘ イドウ
BA20 FE00              CP    00H
BA22 CAABB9            JP    Z,LOPE
BA25 7A                LD    A,D
BA26 FE00              CP    00H
BA28 CAABB9            JP    Z,LOPE
BA2B AF                XOR   A
BA2C 77                LD    (HL),A
BA2D 017900            LD    BC,0079H
BA30 37                SCF
BA31 3F                CCF
BA32 ED42              SBC   HL,BC
BA34 3EEC              LD    A,CHR
```

```
BA36 77                 LD   (HL),A
BA37 1D                 DEC  E
BA38 15                 DEC  D
BA39 C3ABB9             JP   LOPE
                ;
BA3C 7B         UP:     LD   A,E                ;ｳｴ ﾍ ｲﾄﾞｳ
BA3D FE00               CP   00H
BA3F CAABB9             JP   Z,LOPE
BA42 AF                 XOR  A
BA43 77                 LD   (HL),A
BA44 017800             LD   BC,0078H
BA47 37                 SCF
BA48 3F                 CCF
BA49 ED42               SBC  HL,BC
BA4B 3EEC               LD   A,CHR
BA4D 77                 LD   (HL),A
BA4E 1D                 DEC  E
BA4F C3ABB9             JP   LOPE
                ;
BA52 7B         RUP:    LD   A,E                ;ﾐｷﾞｳｴ ﾍ ｲﾄﾞｳ
BA53 FE00               CP   00H
BA55 CAABB9             JP   Z,LOPE
BA58 7A                 LD   A,D
BA59 FE4F               CP   4FH
BA5B CAABB9             JP   Z,LOPE
BA5E AF                 XOR  A
BA5F 77                 LD   (HL),A
BA60 017700             LD   BC,0077H
BA63 37                 SCF
BA64 3F                 CCF
BA65 ED42               SBC  HL,BC
BA67 3EEC               LD   A,CHR
BA69 77                 LD   (HL),A
BA6A 1D                 DEC  E
BA6B 14                 INC  D
BA6C C3ABB9             JP   LOPE
                ;
                ;ｼﾞｶﾝ ﾏﾁ ﾉ ﾙｰﾁﾝ -------------------------------------
                ;
BA6F 0E0F       WAIT:   LD   C,0FH
BA71 06FF       WLP2:   LD   B,0FFH
BA73 05         WLP1:   DEC  B
BA74 C273BA             JP   NZ,WLP1
BA77 0D                 DEC  C
BA78 C271BA             JP   NZ,WLP2
BA7B C9                 RET
                ;
BA7C                    END
```

# 絵描きプログラムを作る

ちょっと応用すると，**リスト3-4**は絵描きのプログラムになります．**リスト3-4**でキャラクタの移動の際に消しているのを，消さないでそのままにしている

だけです．

**●リスト3-5 絵描きプログラム**

```
                    ;***** COLOR TEXT MODE *****
                    ;
                    ;PROGRAM NAME ----> keyinP.e
                    ;
                    ; ｷｰﾆｭｳﾘｮｸ ﾃﾞﾓ ﾌﾟｸﾞﾗﾑ 3(ｴｶｷ    )
                    ;
                    ;.........................................
                    ;
                    ;ｷｬﾗｸﾀｰ ﾉ ｲﾄﾞｳ ｷｰ
                    ;
                    ;              7  8  9
                    ;               \ | /
                    ;              4--●--6
                    ;               / | \         CLS ｷｰ ﾃﾞ ｸﾘｱｽﾙ
                    ;              1  2  3
                    ;                             STOP ｷｰ ﾃﾞ ﾓﾆﾀｰ ﾍ
                    ;.........................................
                    ;
                    ;
B900                START: EQU   0B900H              ;PROGRAM START ADDRESS
E626                MON:   EQU   0E626H              ;ﾓﾆﾀｰ ﾉ ｴﾝﾄﾘ ﾎﾟｲﾝﾄ
6F6B                COLOR: EQU   6F6BH               ;CRT ｺﾝﾄﾛｰﾙ ﾙｰﾁﾝ
00EC                CHR:   EQU   0ECH                ; ｷｬﾗｸﾀｰ ﾉ ｺｰﾄﾞ
                    ;
                    ;
                    ;
                    ;
                           ORG   START
                    ;
                    ;CRT COLOR MODE ---------------------------------
                    ;
B900 011950         START: LD    BC,5019H
B903 3E01                  LD    A,01H
B905 32B2E6                LD    (0E6B2H),A
B908 3E19                  LD    A,19H
B90A 32B3E6                LD    (0E6B3H),A
B90D 3EE8                  LD    A,0E8H
B90F 32B4E6                LD    (0E6B4H),A
B912 3E00                  LD    A,00H
B914 32B8E6                LD    (0E6B8H),A
B917 3EFF                  LD    A,0FFH
B919 32B9E6                LD    (0E6B9H),A
B91C CD6B6F                CALL  COLOR
                    ;
                    ; ﾜｸ ﾄﾞﾘ ｦ ｱｶ ﾆ ｽﾙ -------------------------
                    ;
B91F DD2118F4              LD    IX,0F418H
B923 DD360000              LD    (IX+0),00H          ;ｳｴ ﾉ ｶﾗｰ
B927 DD360148              LD    (IX+1),48H          ;ｱｶ
B92B 0617                  LD    B,17H
B92D 117800                LD    DE,0078H
B930 DD19           LOPI:  ADD   IX,DE
B932 DD360000              LD    (IX+0),00H          ;ﾋﾀﾞﾘ ﾊｼ ﾉ ｶﾗｰ
B936 DD360148              LD    (IX+1),48H          ;ｱｶ
B93A DD360201              LD    (IX+2),01H          ;ﾁｭｳｶﾝ ﾉ ｶﾗｰ
B93E DD360388              LD    (IX+3),088H         ;ｷｬﾗｸﾀｰ ﾉ ｶﾗｰ  ﾐﾄﾞﾘ
```

```
B942 DD36044F          LD    (IX+4),4FH          ;ﾐｷﾞﾊｼ ﾉ ｶﾗｰ
B946 DD360548          LD    (IX+5),48H          ;ｱｶ
B94A 05                DEC   B
B94B C230B9            JP    NZ,LOPI
B94E DD19              ADD   IX,DE
B950 DD360000          LD    (IX+0),00H          ;ｼﾀ ﾉ ｶﾗｰ
B954 DD360148          LD    (IX+1),48H          ;ｱｶ
                  ;
                  ;ｶｰｿﾙ ｦ ｹｽ ﾙｰﾁﾝ ｦ ﾖﾌﾞ ----------------------------------
                  ;
B958 CD8B42            CALL  428BH
                  ;
                  ;
                  ; MAIN ------------------------------------------------
                  ;
B95B 110000            LD    DE,0000H   ;Dﾚｼﾞｽﾀ ﾖｺ ﾉ ｹﾀ ,Eﾚｼﾞｽﾀ ﾀﾃ ﾉ ｷﾞｮｳ
B95E 21C8F3            LD    HL,0F3C8H  ;CRT ﾉ ﾋﾀﾞﾘ ｳｴ ﾉ ｲﾁ
B961 3E27              LD    A,27H
B963 23           LOPM: INC  HL
B964 14                INC   D
B965 3D                DEC   A
B966 C263B9            JP    NZ,LOPM
B969 3E0D              LD    A,0DH
B96B 017800            LD    BC,0078H
B96E 09           LOPN: ADD  HL,BC
B96F 1C                INC   E
B970 3D                DEC   A
B971 C26EB9            JP    NZ,LOPN
                  ;
B974 3EEC              LD    A,CHR          ; ｷｬﾗｸﾀｰ ｦ ﾏﾝﾅｶ ﾆ ﾋｮｳｼﾞ ｽﾙ
B976 77                LD    (HL),A
                  ;
B977 DB00         LOOPK: IN  A,(00H)        ;1 ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B979 47                LD    B,A
B97A E602              AND   02H
B97C CABFB9            JP    Z,LDOWN
                  ;
B97F 78                LD    A,B            ;2 ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B980 E604              AND   04H
B982 CAD9B9            JP    Z,DOWN
                  ;
B985 78                LD    A,B            ;3 ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B986 E608              AND   08H
B988 CAECB9            JP    Z,RDOWN
                  ;
B98B 78                LD    A,B            ;4 ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B98C E610              AND   10H
B98E CA06BA            JP    Z,LEFT
                  ;
B991 78                LD    A,B            ;6 ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B992 E640              AND   40H
B994 CA16BA            JP    Z,RIGHT
                  ;
B997 78                LD    A,B            ;7 ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B998 E680              AND   80H
B99A CA26BA            JP    Z,LUP
                  ;
B99D DB01              IN    A,(01H)        ;8 ｶﾞ ｵｻﾚﾀｶ ｼﾗﾍﾞﾙ
B99F 47                LD    B,A
B9A0 E601              AND   01H
```

```
B9A2 CA43BA          JP    Z,UP
                ;
B9A5 78              LD    A,B           ;9 カ゛ オサレタカ シラヘ゛ル
B9A6 E602            AND   02H
B9A8 CA59BA          JP    Z,RUP
                ;
                ;
B9AB CD96BA     LOPE:  CALL WAIT
                ;
B9AE DB08            IN    A,(08H)       ;CLR キー カ゛ オサレタカ シラヘ゛ル
B9B0 E601            AND   01H
B9B2 CA76BA          JP    Z,CLS
                ;
B9B5 DB09            IN    A,(09H)       ;STOP キー カ゛ オサレタカ シラヘ゛ル
B9B7 E601            AND   01H
B9B9 C277B9          JP    NZ,LOOPK
                ;
B9BC C326E6          JP    MON           ; モニター ヘ モト゛ル
                ;
                ; ● ノ イト゛ウ ルーチン ------------------------------------
                ;
B9BF 7B         LDOWN: LD    A,E           ;ヒタ゛リシタ ヘ イト゛ウ
B9C0 FE18            CP    18H
B9C2 CAABB9          JP    Z,LOPE
B9C5 7A              LD    A,D
B9C6 FE00            CP    00H
B9C8 CAABB9          JP    Z,LOPE
B9CB 00              NOP
B9CC 00              NOP
B9CD 017700          LD    BC,0077H
B9D0 09              ADD   HL,BC
B9D1 3EEC            LD    A,CHR
B9D3 77              LD    (HL),A
B9D4 1C              INC   E
B9D5 15              DEC   D
B9D6 C3ABB9          JP    LOPE
                ;
B9D9 7B         DOWN:  LD    A,E           ;シタ ヘ イト゛ウ
B9DA FE18            CP    18H
B9DC CAABB9          JP    Z,LOPE
B9DF 00              NOP
B9E0 00              NOP
B9E1 017800          LD    BC,0078H
B9E4 09              ADD   HL,BC
B9E5 3EEC            LD    A,CHR
B9E7 77              LD    (HL),A
B9E8 1C              INC   E
B9E9 C3ABB9          JP    LOPE
                ;
B9EC 7B         RDOWN: LD    A,E           ;ミキ゛シタ ヘ イト゛ウ
B9ED FE18            CP    18H
B9EF CAABB9          JP    Z,LOPE
B9F2 7A              LD    A,D
B9F3 FE4F            CP    4FH
B9F5 CAABB9          JP    Z,LOPE
B9F8 00              NOP
B9F9 00              NOP
B9FA 017900          LD    BC,0079H
B9FD 09              ADD   HL,BC
B9FE 3EEC            LD    A,CHR
```

```
BA00 77                 LD    (HL),A
BA01 1C                 INC   E
BA02 14                 INC   D
BA03 C3ABB9             JP    LOPE
                ;
BA06 7A         LEFT:   LD    A,D                ;ﾋﾀﾞﾘ ﾍ ｲﾄﾞｳ
BA07 FE00               CP    00H
BA09 CAABB9             JP    Z,LOPE
BA0C 00                 NOP
BA0D 00                 NOP
BA0E 2B                 DEC   HL
BA0F 3EEC               LD    A,CHR
BA11 77                 LD    (HL),A
BA12 15                 DEC   D
BA13 C3ABB9             JP    LOPE
                ;
BA16 7A         RIGHT:  LD    A,D                ;ﾐｷﾞ ﾍ ｲﾄﾞｳ
BA17 FE4F               CP    4FH
BA19 CAABB9             JP    Z,LOPE
BA1C 00                 NOP
BA1D 00                 NOP
BA1E 23                 INC   HL
BA1F 3EEC               LD    A,CHR
BA21 77                 LD    (HL),A
BA22 14                 INC   D
BA23 C3ABB9             JP    LOPE
                ;
BA26 7B         LUP:    LD    A,E                ;ﾋﾀﾞﾘｳｴ ﾍ ｲﾄﾞｳ
BA27 FE00               CP    00H
BA29 CAABB9             JP    Z,LOPE
BA2C 7A                 LD    A,D
BA2D FE00               CP    00H
BA2F CAABB9             JP    Z,LOPE
BA32 00                 NOP
BA33 00                 NOP
BA34 017900             LD    BC,0079H
BA37 37                 SCF
BA38 3F                 CCF
BA39 ED42               SBC   HL,BC
BA3B 3EEC               LD    A,CHR
BA3D 77                 LD    (HL),A
BA3E 1D                 DEC   E
BA3F 15                 DEC   D
BA40 C3ABB9             JP    LOPE
                ;
BA43 7B         UP:     LD    A,E                ;ｳｴ ﾍ ｲﾄﾞｳ
BA44 FE00               CP    00H
BA46 CAABB9             JP    Z,LOPE
BA49 00                 NOP
BA4A 00                 NOP
BA4B 017800             LD    BC,0078H
BA4E 37                 SCF
BA4F 3F                 CCF
BA50 ED42               SBC   HL,BC
BA52 3EEC               LD    A,CHR
BA54 77                 LD    (HL),A
BA55 1D                 DEC   E
BA56 C3ABB9             JP    LOPE
                ;
BA59 7B         RUP:    LD    A,E                ;ﾐｷﾞｳｴ ﾍ ｲﾄﾞｳ
```

```
BA5A FE00               CP    00H
BA5C CAABB9             JP    Z,LOPE
BA5F 7A                 LD    A,D
BA60 FE4F               CP    4FH
BA62 CAABB9             JP    Z,LOPE
BA65 00                 NOP
BA66 00                 NOP
BA67 017700             LD    BC,0077H
BA6A 37                 SCF
BA6B 3F                 CCF
BA6C ED42               SBC   HL,BC
BA6E 3EEC               LD    A,CHR
BA70 77                 LD    (HL),A
BA71 1D                 DEC   E
BA72 14                 INC   D
BA73 C3ABB9             JP    LOPE
                ;
BA76 D5         CLS:    PUSH  DE              ;CRT ノ クリア
BA77 FD21C8F3           LD    IY,0F3C8H
BA7B 112800             LD    DE,0028H
BA7E AF                 XOR   A
BA7F 0E19               LD    C,19H
BA81 0650       CLLP2:  LD    B,50H
BA83 FD7700     CLLP1:  LD    (IY),A
BA86 FD23               INC   IY
BA88 05                 DEC   B
BA89 C283BA             JP    NZ,CLLP1
BA8C FD19               ADD   IY,DE
BA8E 0D                 DEC   C
BA8F C281BA             JP    NZ,CLLP2
BA92 D1                 POP   DE
BA93 C300B9             JP    START
                ;
                ;
                ;ジカン マチ ノ ルーチン -----------------------------------
                ;
BA96 0E0F       WAIT:   LD    C,0FH
BA98 06FF       WLP2:   LD    B,0FFH
BA9A 05         WLP1:   DEC   B
BA9B C29ABA             JP    NZ,WLP1
BA9E 0D                 DEC   C
BA9F C298BA             JP    NZ,WLP2
BAA2 C9                 RET
                ;
BAA3                    END
```

これでキー入力の説明を終りますが，理解できたでしょうか？　この章を読み終えると楽なものですね．次章はグラフィックスの世界に入ります．

# 第4章 グラフィックスの世界

## グラフィックス・パターンの作り方と移動方法

今のパソコンには，グラフィックス機能がついているのが当たり前になっています．グラフィックス機能は美しいのでパソコンの魅力のひとつとなっていますが，BASIC 命令で使用すると遅くてイライラします．BASIC 命令だけで，グラフィックスを利用したリアルタイム・ゲームを作るのは事実上不可能で，当然，スピードの速いマシン語を使わなければなりません．

そこでこの章と次の章で，皆さんをグラフィックスの世界に招待します．この章では何ビットかの集まりを1ブロックとして，ブロック単位でパターン（ヘリコプター）を移動させてみましょう．

*Section 1*

# グラフィックスの原理

Graphics

## ビット・マップ法によるグラフィックス

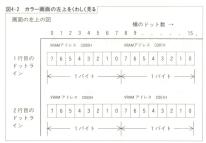
TV（CRT を含む）に描き出される画像は，点の集まりから出来ていることは知っていると思います．この画像を作る点を画素（ドット）といいます．この画素が多いほど鮮明な画像を描くことができることになります．PC-8801 mkIISR では**図4-1**のようになっています．

**図4-1　PC-8801mkIISRのグラフィックス・ドット構成**

このように，カラー画面ではモノクロ画面の半分の表示能力しかありませんが，ゲームなどに使用するには十分です．

## ■カラーの画面構成をくわしく見ると…

もう少しカラーの画面構成をくわしくみることにしましょう．カラー画面の左上の部分は図4-2のようになっています．

図4-2　カラー画面の左上をくわしく見る

このように，各ビットの表し方は8ビットを1つのブロックとして扱っています．つまり，1バイトを1つの単位としていることになります．このことは8ビット・マシンに都合の良いことです．

## ■1ドットが1ビットに対応するビット・マップ法

1ドットは，1ビットに対応しています．このことを，ビット・マップ法と言います．

ビット・マップ法 → 1ドットが1ビットに対応している VRAM 構成
ビットが1のとき………ドットが表示される
ビットが0のとき………ドットが消去される

この方法はグラフィックス・パターンのデータを作るのに，比較的楽な方法です．高級タイプのパソコンはこの方法が多いようです．

したがって，実際のグラフィックスのパターンを作るときは，8ビットごとに1バイトのデータに変換してやる必要があります．このグラフィックス・パターンのデータ変換は手作業でやらなければなりません．

### ■カラーモードにおける G-VRAM の番地

8ビットを1単位としているのですから，この1単位ごとに VRAM の番地が割り振られています．横の番地数は，

640ドット÷8ビット＝80番地

となります．グラフィックスの VRAM 番地(G-VRAM)は**図4-3**のようになります．ただし，テキストモードのときのように，アトリビュートエリアはありません．

図4-3　カラーモード時のG-VRAMの番地

| 1 | 2 | ・・・・・・・・・・・・・・・・ | 79 | 80バイト |
|---|---|---|---|---|
| C000 | C001 | ・・・・・・・・・・・・・・・・ | C04E | C04F |
| C050 | C051 | ・・・・・・・・・・・・・・・・ | C0AE | C0AF |
| ・・・ | ・・・ | ・・・・・・・・・・・・・・・・ | ・・・ | ・・・ |
| FDE0 | FDE1 | ・・・・・・・・・・・・・・・・ | FE2E | FE2F |
| FE30 | FE31 | ・・・・・・・・・・・・・・・・ | FE7E | FE7F |

このように横80バイトで縦が200ラインで200バイトになりますから，

80バイト×200バイト＝16000バイト（16Kバイト）

つまり，600ドット×200ドットで16Kバイトのメモリ空間が必要になるわけです．

## グラフィックス・パターンをカラーにするには

グラフィックス・パターンをカラーにするには，少し複雑になります．グラフィックスでは，カラーテレビの原理をそのまま使用しています．カラーテレビはR（赤），G（緑），B（青）の3原色を微妙にコントロールしていろいろな色を出しています．

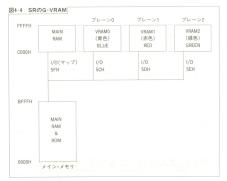
グラフィックスのカラーも原理は同じでR，G，Bに対応する VRAM を持っています．1つの色について16Kバイト必要ですから，カラーにするためには，

16Kバイト×3カラー分＝48Kバイト

必要となります．

## ■バンク切り替えで48Kバイトの G-VRAM

48Kバイトもメモリを必要とするので，メイン・メモリのワークエリアが減らないように，バンク切り替えという方法をとってメイン・メモリが減らないように工夫しています．このバンク切り替えという方法は Z-80 CPU を使用しているパソコンでは，ごく当たり前のことです．このバンクの切り替えは“OUT”命令を使用することによって，ソフトウエア的に切り替えができます．このカラー化のための VRAM をプレーンとも呼びます．PC-8801mkIISR のプレーンは**図4-4**のようになっています．

図4-4　SRのG-VRAM

## ■3プレーンで8色出せる

どのプレーンにグラフィックス・パターンを書き込むかによって，グラフィックス・パターンに色がつきます．3プレーンありますので，プレーンの組み合わせで8色まで出ます．プレーンと色の関係は**表4-1**のようになります．

もし，グラフィックス・パターンを白にするなら3プレーンにデータを書き込まなくてはなりません．

表4-1　プレーンと色の関係

| プレーン0<br>B | プレーン1<br>R | プレーン2<br>G | 色 |
|---|---|---|---|
| 0 | 0 | 0 | 黒 |
| 1 | 0 | 0 | 青 |
| 0 | 1 | 0 | 赤 |
| 1 | 1 | 0 | 紫 |
| 0 | 0 | 1 | 緑 |
| 1 | 0 | 1 | 水色 |
| 0 | 1 | 1 | 黄 |
| 1 | 1 | 1 | 白 |

## カラープレーンを切り替えるには

マシン語で直接G-VRAMを操作する際，バンク切り替えに“OUT”命令を使用するのは良いのですが，その際には次の細かい条件を守って使用しなければなりません．

**G-VRAM操作上の注意**

1．G-VRAMのアクセス・ルーチンは，必ずC000H番地以前になくてはなりません．
2．バンク切り替えの前に，必ず“DI”命令をしなくてはなりません．割り込みによって起こる暴走を防ぐためです．
3．G-VRAMのアクセスが終ったならば，バンク切り替えでメイン･RAMに必ず切り替え，“EI”命令を実行しなければなりません．

この条件はどんなことがあっても必ず守ってください．もし，この条件のひとつでも守らないと暴走を起こします．

### ■割り込みをコントロールするDI，EI命令

PC-8801mkIISRのCRTコントローラは割り込み処理をしているので，ハードウェアで定期的にシステム･RAM(E600H番地以降)を読み取りにきます．このときに，正しいデータでないと暴走を起こします．暴走しないように，シ

ステム・RAM以外をアクセスしているときは割り込みが起こらないようにしなければなりません．割り込みをコントロールしているのが，この2つの命令なのです．

DI命令……………………割り込み禁止
EI命令……………………割り込み許可

## ■カラープレーン切り替えの手順

実際の手順は次のように行います，

```
      ⋮
      DI
      OUT (GRAM), A ；プレーンのI/Oアドレス
      ⋮              ；プレーンの書き込みルーチン
      OUT (5FH), A  ；メイン・メモリにする
      EI
      ⋮
      ⋮
```

なお，このAレジスタのデータは意味のないものです．したがって，どんなデータでもかまいません．

このようになりますが，実際のサンプル・プログラムをみれば理解できるでしょう．

# *Section 2*

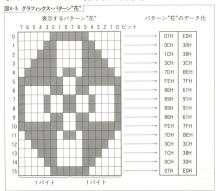
# グラフィックスのプログラムを作る

Graphics

## グラフィックス・パターン"花"の表示

グラフィックスの原理が理解できたところで，実際のサンプルプログラムを作ってみましょう．

まず初めは，簡単なグラフィックス・パターンの表示からです．**図4-5**のようなグラフィックス・パターンを表示してみましょう．

**図4-5 グラフィックス・パターン"花"**

このままのパターンでも良いのですが，パターンを移動したりするときに，前のパターンの残りが出ないようにするために，パターンのまわりを00Hで囲んでおくと，プログラミングが楽になります．では，パターンをデータ化してみましょう（**表4-2**）．

このように，表示しようとするグラフィックス・パターンは，すべてデータ化しなければなりません．複雑なグラフィックス・パターンほど，データ化がやっかいです．パターン"花"のデータ化の手順は**図4-6**のようになります．

なお，わざわざ2進数の0，1で表す必要はありません．直接，16進数でデータ化してしまえば良いのです．

**表4-2　プログラム上のパターンのデータ化**

| 00 | 00 | 00 | 00 |
|---|---|---|---|
| 00 | 07 | E0 | 00 |
| 00 | 0C | 30 | 00 |
| 00 | 1C | 38 | 00 |
| 00 | 3C | 3C | 00 |
| 00 | 7D | BE | 00 |
| 00 | FE | 7F | 00 |
| 00 | 86 | 61 | 00 |
| 00 | 89 | 91 | 00 |
| 00 | 89 | 91 | 00 |
| 00 | 86 | 61 | 00 |
| 00 | FE | 7F | 00 |
| 00 | 7D | BE | 00 |
| 00 | 3C | 3C | 00 |
| 00 | 1C | 38 | 00 |
| 00 | 0C | 30 | 00 |
| 00 | 07 | E0 | 00 |
| 00 | 00 | 00 | 00 |

この00Hのデータは，パターンがどちらの方向に移動しても，前のパターンを消すのに必要．

**図4-6　パターン"花"のデータ化手順**

## CRT中央に"花"を表示するプログラムの作成

それでは，CRT上の中央にパターン"花"を表示するプログラムを考えてみ

ましょう．パターン“花”は4バイト×18ラインですから，1ラインの4バイトをカラープレーンに書き込んだ後は，下のラインを表示しなければなりません．表示位置を下のラインまで下げるには，現在の番地に50H 番地足します．テキストモードでは78H を足していましたが，グラフィックスモードではアトリビュートエリアがないので，80桁分の50H で良いのです．

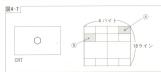
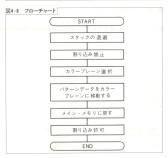
実際には**図4-7**のとおり，A点からB点に行かなければならないので，4バイト分引く必要があります．A点からB点に移動するときに加えるデータは，

0050H－0004H＝004CH

となります．

フローチャートを作りましょう．**図4-8**のようになります．

**図4-7**

**図4-8　フローチャート**

## ■スタック・エリアを広げる方法

グラフィックスのプログラムでは，スタック・ポインタをよく使用するので，自分のプログラム用にスタック・エリアを広げておかなければなりません．ただし，現在のスタック・ポインタの位置を保存しておく必要があります．そうしないと暴走してしまいます．スタック・エリアを広げる方法を示します．

**スタック・ポインタの設定**

```
        ∫
LD(SPSAVE), SP………現在のスタック・ポインタ(SP)の位置をSPSAVEと
                     名のついたエリアに保存
LD SP, STACK…………新たにSTACKの位置にSPを設定
        ∫
LD SP, (SPSAVE)………モニタへ戻るのでSPを以前に保存しておいた位置にする
JP MON…………………………モニタへ戻る
        ∫
SPSAVE：DS 02H………元スタックデータ
        DS 60…………スタックのエリア
STACK：……………………この位置から番地の小さい方へ60バイト（30レベル）のスタック・エリアを確保した
```

## ■"花"のプログラム

さて，プログラムは次のようになります．

**●リスト4-1 "花"表示プログラム**

```
                    ;*************************************************
                    ;
                    ;GRPH--1
                    ;
                    ;PROGRAM NAME ---> Grph1.e
                    ;
                    ;*************************************************
                    ;
B900                START:  EQU   0B900H
E000                GRAMS:  EQU   0E000H
E626                MON:    EQU   0E626H
005C                GRAM0:  EQU   5CH
005D                GRAM1:  EQU   5DH
005E                GRAM2:  EQU   5EH
005F                MRAM:   EQU   5FH
                    ;
                            ORG   START
                    ;
B900 ED7374B9               LD    (SPSAVE).SP          ;ｹﾞﾝｻﾞｲ ﾉ ｽﾀｯｸ ﾀｲﾋ
B904 319EB9                 LD    SP.STACK             ;ｱﾗﾀﾆ ｽﾀｯｸ ｦ ｾｯﾃｲ
                    ;
B907 2100E0                 LD    HL.GRAMS
```

```
B90A 1129B9             LD    DE,CHRDATA
                ;
B90D F3                 DI
                ;
B90E D35D               OUT   (GRAM1),A
                ;
                ;
B910 0E12               LD    C,12H
B912 0604       LOP2:   LD    B,04H
B914 1A         LOP1:   LD    A,(DE)
B915 77                 LD    (HL),A
B916 23                 INC   HL
B917 13                 INC   DE
B918 05                 DEC   B
B919 C214B9             JP    NZ,LOP1
B91C C5                 PUSH  BC
B91D 014C00             LD    BC,004CH
B920 09                 ADD   HL,BC
B921 C1                 POP   BC
B922 0D                 DEC   C
B923 C212B9             JP    NZ,LOP2
                ;
B926 D35F               OUT   (MRAM),A
B928 FB                 EI
                ;
B929 00         CHRDATADB     00
B92A 00                 DB    00
B92B 00                 DB    00
B92C 00                 DB    00
B92D 00                 DB    00
B92E 07                 DB    07H
B92F E0                 DB    0E0H
B930 00                 DB    00
B931 00                 DB    00
B932 0C                 DB    0CH
B933 30                 DB    30H
B934 00                 DB    00
B935 00                 DB    00
B936 1C                 DB    1CH
B937 38                 DB    38H
B938 00                 DB    00
B939 00                 DB    00
B93A 3C                 DB    3CH
B93B 3C                 DB    3CH
B93C 00                 DB    00
B93D 00                 DB    00
B93E 7D                 DB    7DH
B93F BE                 DB    0BEH
B940 00                 DB    00
B941 00                 DB    00
B942 FE                 DB    0FEH
B943 7F                 DB    7FH
B944 00                 DB    00
B945 00                 DB    00
B946 86                 DB    86H
B947 61                 DB    61H
B948 00                 DB    00
B949 00                 DB    00
B94A 89                 DB    89H
B94B 91                 DB    91H
```

```
B94C 00                 DB    00
B94D 00                 DB    00
B94E 89                 DB    89H
B94F 91                 DB    91H
B950 00                 DB    00
B951 00                 DB    00
B952 86                 DB    86H
B953 61                 DB    61H
B954 00                 DB    00
B955 00                 DB    00
B956 FE                 DB    0FEH
B957 7E                 DB    7EH
B958 00                 DB    00
B959 00                 DB    00
B95A 7D                 DB    7DH
B95B BE                 DB    0BEH
B95C 00                 DB    00
B95D 00                 DB    00
B95E 3C                 DB    3CH
B95F 3C                 DB    3CH
B960 00                 DB    00
B961 00                 DB    00
B962 1C                 DB    1CH
B963 38                 DB    38H
B964 00                 DB    00
B965 00                 DB    00
B966 0C                 DB    0CH
B967 30                 DB    30H
B968 00                 DB    00H
B969 00                 DB    00
B96A 07                 DB    07H
B96B E0                 DB    0E0H
B96C 00                 DB    00
B96D 00                 DB    00
B96E 00                 DB    00
B96F 00                 DB    00
B970 00                 DB    00
                ;
B971 C326E6             JP    MON
                ;
B974            SPSAVE:DS     02H                ;ﾓﾄ ﾉ ｽﾀｯｸ ﾃﾞｰﾀ
B976                    DS    40
B99E            STACK:                           ;ｱﾀﾗｼｸ ｾｯﾃｲｼﾀ ｽﾀｯｸ ｴﾘｱ
                ;
B99E                    END
```

## ヘリコプターを表示する

パターン“花”は16ドット×16ドットの正方形の表示だったのですが，CRTの関係上，縦長の表示パターンになってしまいました．もし，正方形型に表示したいなら，横のドット数を2倍にしてやらなければなりません．

ここで作るプログラムは，パターン“花”と全く同じ方法で作るものですが，

もう少し複雑にして，後のプログラムでも使用できるようなものにしてみましょう．ここでは，正方形型になるようにしてあります．

表示するのは**図4-9**のようなヘリコプターのパターンです．

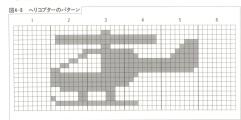



**図4-9　ヘリコプターのパターン**

ヘリコプターのパターン・データは**表4-3**のとおりです．

**表4-3　ヘリコプターのパターン・データ**

| |
|---|
| 00H, 00H, 00H, 00H, 00H, 00H |
| 00H, 00H, 18H, 00H, 00H, 00H |
| 00H, 07FH, 0FFH,0FEH, 00H, 00H |
| 00H,07FH, 0FFH,0FEH, 07H, 00H |
| 00H, 00H, 18H, 00H, 0FH, 00H |
| 00H, 01H, 0FFH, 0C0H, 1FH, 00H |
| 00H, 03H, 1FH, 0E0H, 3FH, 00H |
| 00H, 0CH, 1FH, 0FFH, 0FFH, 00H |
| 00H, 30H, 1FH, 0FFH, 0FFH, 00H |
| 00H, 0C0H, 1FH, 0FFH, 0FEH, 00H |
| 00H, 0FFH, 0FFH, 0E0H, 00H, 00H |
| 00H, 0FFH, 0FFH, 0C0H, 00H, 00H |
| 00H, 0FFH, 0FFH, 80H, 00H, 00H, |
| 00H, 3FH, 0FCH, 00H, 00H, 00H |
| 00H, 03H, 0CH, 00H, 00H, 00H |
| 00H, 83H, 0CH, 00H, 00H, 00H |
| 00H, 7FH, 0FFH, 0E0H, 00H, 00H, |
| 00H, 00H, 00H, 00H, 00H, 00H |

プログラムは**リスト4-2**のようになります．

**●リスト4-2 ヘリコプターを表示する**

```
                ;*************************************************
                ;
                ;GRPH--2
                ;
                ;PROGRAM NAME ---> Grph2.e
                ;
                ;*************************************************
                ;
B900            START:  EQU   0B900H
E000            GRAMS:  EQU   0E000H
E626            MON:    EQU   0E626H
005C            GRAM0:  EQU   5CH
005D            GRAM1:  EQU   5DH
005E            GRAM2:  EQU   5EH
005F            MRAM:   EQU   5FH
                ;
                        ORG   START
                ;
B900 ED7398B9           LD    (SPSAVE),SP        ;ｹﾞﾝｻﾞｲ ﾉ ｽﾀｯｸ ﾉ ﾀｲﾋ
B904 31D6B9             LD    SP,STACK           ;ｱﾗﾀﾆ ｽﾀｯｸ  ｦ ｾｯﾃｲ
                ;
B907 2100E0             LD    HL,GRAMS
B90A 112CB9             LD    DE,CHRDATA
                ;
B90D F3                 DI
                ;
B90E D35E               OUT   (GRAM2),A
                ;
                ;
B910 0E12               LD    C,12H
B912 0606       LOP2:   LD    B,06H
B914 1A         LOP1:   LD    A,(DE)
B915 77                 LD    (HL),A
B916 23                 INC   HL
B917 13                 INC   DE
B918 05                 DEC   B
B919 C214B9             JP    NZ,LOP1
B91C C5                 PUSH  BC
B91D 014A00             LD    BC,004AH
B920 09                 ADD   HL,BC
B921 C1                 POP   BC
B922 0D                 DEC   C
B923 C212B9             JP    NZ,LOP2
                ;
B926 D35F               OUT   (MRAM),A
B928 FB                 EI
                ;
B929 C326E6             JP    MON
                ;
                ;
                ;
B92C 00000000   CHRDATADB     00H,00H,00H,00H,00H,00H
B930 0000
B932 00001800           DB    00H,00H,18H,00H,00H,00H
B936 0000
B938 007FFFFE           DB    00H,7FH,0FFH,0FEH,00H,00H
```

```
B93C 0000
B93E 007FFFFE         DB     00H,7FH,0FFH,0FEH,07H,00H
B942 0700
B944 00001800         DB     00H,00H,18H,00H,0FH,00H
B948 0F00
B94A 0001FFC0         DB     00H,01H,0FFH,0C0H,1FH,00H
B94E 1F00
B950 00031FE0         DB     00H,03H,1FH,0E0H,3FH,00H
B954 3F00
B956 000C1FFF         DB     00H,0CH,1FH,0FFH,0FFH,00H
B95A FF00
B95C 00301FFF         DB     00H,30H,1FH,0FFH,0FFH,00H
B960 FF00
B962 00C01FFF         DB     00H,0C0H,1FH,0FFH,0FEH,00H
B966 FE00
B968 00FFFFE0         DB     00H,0FFH,0FFH.0E0H,00H,00H
B96C 0000
B96E 00FFFFC0         DB     00H,0FFH,0FFH,0C0H,00H,00H
B972 0000
B974 00FFFF80         DB     00H,0FFH,0FFH,80H,00H,00H
B978 0000
B97A 003FFC00         DB     00H,3FH,0FCH,00H,00H,00H
B97E 0000
B980 00030C00         DB     00H,03H,0CH,00H,00H,00H
B984 0000
B986 00830C00         DB     00H,83H,0CH,00H,00H,00H
B98A 0000
B98C 007FFFE0         DB     00H,7FH,0FFH,0E0H,00H,00H
B990 0000
B992 00000000         DB     00H,00H,00H,00H,00H,00H
B996 0000
                 :
                 :
B998             SPSAVE:DS     02H
B99A                    DS     60
B9D6             STACK:
                 :
B9D6                    END
```

## ヘリコプターの翼を回転させる

ヘリコプターの形を表示するだけでは面白くありません．そこで，ヘリコプターの翼を回転させてみることにしましょう．では，どのようにしたら，ヘリコプターの翼が回転しているように見えるでしょうか．**図4-10** (P.174) のパターンを繰り返せば良いのです．

A，B，C，Dの４つのパターンを定期的に繰り返してCRT上に表示すれば，回転しているように見えます．B，Dは同じパターンなので３つのパターンで済みます．

図4-10 ヘリコプターの翼を回転させる

プログラムも簡単です．A，B，C，Dのパターン・データを順番に同じところに表示させるだけです．1つのパターンから他のパターンに移るときに時間待ちしていますが，この待ち時間を調整することによって回転する速さをコントロールできます．**リスト4-3**にプログラムを示します．

**●リスト4-3 ヘリコプターの翼を回転させる**

```
                   ;*************************************************
                   ;
                   ;GRPH--3 ﾍﾘｺﾌﾟﾀｰ ﾉ ﾊﾈ ﾉ ｶｲﾃﾝ
                   ;
                   ;PROGRAM NAME ---> Grph3.e
                   ;
                   ;*************************************************
                   ;
B900               START:  EQU   0B900H
E000               GRAMS:  EQU   0E000H
428B               CLS:    EQU   428BH
E626               MON:    EQU   0E626H
005C               GRAM0:  EQU   5CH
005D               GRAM1:  EQU   5DH
005E               GRAM2:  EQU   5EH
005F               MRAM:   EQU   5FH
                   ;
                           ORG   START
                   ;
B900 CD8B42                CALL  CLS                 ;ｶｰｿﾙ OFF
                   ;
B903 ED73C1BA              LD    (SPSAVE),SP         ;ｹﾞﾝｻﾞｲ ﾉ ｽﾀｯｸ ﾀｲﾋ
B907 31FFBA                LD    SP,STACK            ;ｱﾗﾀﾆ ｽﾀｯｸ ﾉ ｾｯﾃｲ
                   ;
B90A 0603          MAIN:   LD    B,03H
B90C 78            KAITEN:LD     A,B
B90D FE03                  CP    03H
B90F C21FB9                JP    NZ,CD2
B912 2100E0                LD    HL,GRAMS
B915 117DB9                LD    DE,CDATA1
B918 CD5EB9                CALL  DISP
```

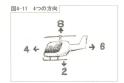
```
B91B 05               DEC  B
B91C C349B9           JP   WAIT
               ;
B91F 78        CD2:   LD   A,B
B920 FE02             CP   02H
B922 C232B9           JP   NZ,CD3
B925 2100E0           LD   HL,GRAMS
B928 11E9B9           LD   DE,CDATA2
B92B CD5EB9           CALL DISP
B92E 05               DEC  B
B92F C349B9           JP   WAIT
               ;
B932 2100E0    CD3:   LD   HL,GRAMS
B935 1155BA           LD   DE,CDATA3
B938 CD5EB9           CALL DISP
B93B CD4FB9           CALL WAT
B93E 2100E0           LD   HL,GRAMS
B941 11E9B9           LD   DE,CDATA2
B944 CD5EB9           CALL DISP
B947 0603             LD   B,03H
               ;
B949 CD4FB9    WAIT:  CALL WAT
               ;
B94C C30CB9           JP   KAITEN
               ;
B94F D5        WAT:   PUSH DE                    ;ジカン マチ
B950 160F             LD   D,0FH
B952 1EFF      W2:    LD   E,0FFH
B954 1D        W1:    DEC  E
B955 C254B9           JP   NZ,W1
B958 15               DEC  D
B959 C252B9           JP   NZ,W2
B95C D1               POP  DE
B95D C9               RET
               ;
               ;
B95E F3        DISP:  DI                         ;グラフ ノ ヒョウジ ルーチン ワリコミ NO
B95F D35E             OUT  (GRAM2),A             ;カラー ノ イロ グリーン
B961 C5               PUSH BC
B962 0E12             LD   C,12H
B964 0606      LOP2:  LD   B,06H
B966 1A        LOP1:  LD   A,(DE)
B967 77               LD   (HL),A
B968 23               INC  HL
B969 13               INC  DE
B96A 05               DEC  B
B96B C266B9           JP   NZ,LOP1
B96E C5               PUSH BC
B96F 014A00           LD   BC,004AH
B972 09               ADD  HL,BC
B973 C1               POP  BC
B974 0D               DEC  C
B975 C264B9           JP   NZ,LOP2
B978 C1               POP  BC
B979 D35F             OUT  (MRAM),A              ;メイン メモリー ニ スル
B97B FB               EI                         ;ワリコミ OK
               ;
B97C C9               RET
               ;
               ;
```

```
                  ;
B97D 00000000 CDATA1:DB    00H,00H,00H,00H,00H,00H
B981 0000
B983 00001800        DB    00H,00H,18H,00H,00H,00H
B987 0000
B989 00FFFFFF        DB    00H,0FFH,0FFH,0FFH,00H,00H
B98D 0000
B98F 00FFFFFF        DB    00H,0FFH,0FFH,0FFH,07H,00H
B993 0700
B995 00001800        DB    00H,00H,18H,00H,0FH,00H
B999 0F00
B99B 0001FFC0        DB    00H,01H,0FFH,0C0H,1FH,00H
B99F 1F00
B9A1 00031FE0        DB    00H,03H,1FH,0E0H,3FH,00H
B9A5 3F00
B9A7 000C1FFF        DB    00H,0CH,1FH,0FFH,0FFH,00H
B9AB FF00
B9AD 00301FFF        DB    00H,30H,1FH,0FFH,0FFH,00H
B9B1 FF00
B9B3 00C01FFF        DB    00H,0C0H,1FH,0FFH,0FEH,00H
B9B7 FE00
B9B9 00FFFFE0        DB    00H,0FFH,0FFH,0E0H,00H,00H
B9BD 0000
B9BF 00FFFFC0        DB    00H,0FFH,0FFH,0C0H,00H,00H
B9C3 0000
B9C5 00FFFF80        DB    00H,0FFH,0FFH,80H,00H,00H
B9C9 0000
B9CB 003FFC00        DB    00H,3FH,0FCH,00H,00H,00H
B9CF 0000
B9D1 00030C00        DB    00H,03H,0CH,00H,00H,00H
B9D5 0000
B9D7 00C30C00        DB    00H,0C3H,0CH,00H,00H,00H
B9DB 0000
B9DD 007FFFE0        DB    00H,7FH,0FFH,0E0H,00H,00H
B9E1 0000
B9E3 00000000        DB    00H,00H,00H,00H,00H,00H
B9E7 0000
                  ;
                  ;
                  ;
B9E9 00000000 CDATA2:DB    00H,00H,00H,00H,00H,00H
B9ED 0000
B9EF 00001800        DB    00H,00H,18H,00H,00H,00H
B9F3 0000
B9F5 0000FF00        DB    00H,00H,0FFH,00H,00H,00H
B9F9 0000
B9FB 0000FF00        DB    00H,00H,0FFH,00H,07H,00H
B9FF 0700
BA01 00001800        DB    00H,00H,18H,00H,0FH,00H
BA05 0F00
BA07 0001FFC0        DB    00H,01H,0FFH,0C0H,1FH,00H
BA0B 1F00
BA0D 00031FE0        DB    00H,03H,1FH,0E0H,3FH,00H
BA11 3F00
BA13 000C1FFF        DB    00H,0CH,1FH,0FFH,0FFH,00H
BA17 FF00
BA19 00301FFF        DB    00H,30H,1FH,0FFH,0FFH,00H
BA1D FF00
BA1F 00C01FFF        DB    00H,0C0H,1FH,0FFH,0FEH,00H
BA23 FE00
BA25 00FFFFE0        DB    00H,0FFH,0FFH,0E0H,00H,00H
```

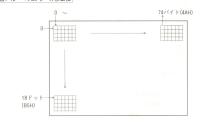
```
BA29 0000
BA2B 00FFFFC0           DB      00H,0FFH,0FFH,0C0H,00H,00H
BA2F 0000
BA31 00FFFF80           DB      00H,0FFH,0FFH,80H,00H,00H
BA35 0000
BA37 003FFC00           DB      00H,3FH,0FCH,00H,00H,00H
BA3B 0000
BA3D 00030C00           DB      00H,03H,0CH,00H,00H,00H
BA41 0000
BA43 00830C00           DB      00H,83H,0CH,00H,00H,00H
BA47 0000
BA49 007FFFE0           DB      00H,7FH,0FFH,0E0H,00H,00H
BA4D 0000
BA4F 00000000           DB      00H,00H,00H,00H,00H,00H
BA53 0000
                  ;
BA55 00000000 CDATA3:DB      00H,00H,00H,00H,00H,00H
BA59 0000
BA5B 00001800           DB      00H,00H,18H,00H,00H,00H
BA5F 0000
BA61 00001800           DB      00H,00H,18H,00H,00H,00H
BA65 0000
BA67 00001800           DB      00H,00H,18H,00H,07H,00H
BA6B 0700
BA6D 00001800           DB      00H,00H,18H,00H,0FH,00H
BA71 0F00
BA73 0001FFC0           DB      00H,01H,0FFH,0C0H,1FH,00H
BA77 1F00
BA79 00031FE0           DB      00H,03H,1FH,0E0H,3FH,00H
BA7D 3F00
BA7F 000C1FFF           DB      00H,0CH,1FH,0FFH,0FFH,00H
BA83 FF00
BA85 00301FFF           DB      00H,30H,1FH,0FFH,0FFH,00H
BA89 FF00
BA8B 00C01FFF           DB      00H,0C0H,1FH,0FFH,0FEH,00H
BA8F FE00
BA91 00FFFFE0           DB      00H,0FFH,0FFH,0E0H,00H,00H
BA95 0000
BA97 00FFFFC0           DB      00H,0FFH,0FFH,0C0H,00H,00H
BA9B 0000
BA9D 00FFFF80           DB      00H,0FFH,0FFH,80H,00H,00H
BAA1 0000
BAA3 003FFC00           DB      00H,3FH,0FCH,00H,00H,00H
BAA7 0000
BAA9 00030C00           DB      00H,03H,0CH,00H,00H,00H
BAAD 0000
BAAF 00830C00           DB      00H,83H,0CH,00H,00H,00H
BAB3 0000
BAB5 007FFFE0           DB      00H,7FH,0FFH,0E0H,00H,00H
BAB9 0000
BABB 00000000           DB      00H,00H,00H,00H,00H,00H
BABF 0000
                  ;
                  ;
BAC1          SPSAVE:DS      2
BAC3                 DS      60
BAFF          STACK:
                  ;
BAFF                 END
```

# ヘリコプターを移動させる

ヘリコプターをキーで移動させるプログラムを作ってみましょう．まずは簡単に4方向（上，下，左，右）に移動させてみましょう（**図4-11**）．

**図4-11　4つの方向**

## ■難しそうだけど実は簡単

難しそうにみえますが，楽にできます．というのは，以前第3章3節でキー入力のプログラムを作っているからです．このプログラムを一部手直しして，ヘリコプター移動のプログラムを作ります．以前のプログラムでは，移動キャラクタが1バイト・データでしたが，今回のは108バイトのキャラクタになる点だけが違います．1バイト・データを表示するのも，108バイトのデータを表示するのも，アルゴリズム自体には変りありません．1バイト・データを表示す

**図4-12　ヘリコプターの移動幅**

る代りに，108バイトを表示するサブルーチンを呼ぶだけのことです．

ヘリコプターの移動幅は，

横の移動幅　80バイト－ヘリコプターの横のバイト数（6バイト）
＝74バイト（4AH）

縦の移動幅　200ドット－ヘリコプターの縦のドット数（18ドット）
＝182ドット（B6H）……プログラム上ではB5H

となります．ヘリコプターの回転翼はいつも回転していなければならないので，キー入力の部分はサブルーチンとなります．

## ■ヘリコプターを４方向に移動させる

では，ヘリコプターを4方向に移動させるプログラムを**リスト4-4**に示します．このプログラムでは，上下方向はドット単位で移動しているのでスムーズに移動しますが，横方向は1バイト（8ビット）単位の移動なので，動きが雑になっています(**図4-13**)．

図4-13　ヘリコプターの移動のようす

なお，横方向をスムーズに移動させるプログラムは第5章で説明しますので，そちらを御覧ください．

●リスト4-4 ヘリコプターの移動(4方向)

```
                    ;*******************************************************
                    ;
                    ;GRPH--4 ヘリコプ゚ター ノ ハネ ノ カイテン ト キーニュウリョク
                    ;
                    ;PROGRAM NAME ---> Grph4.e
                    ;
                    ;*******************************************************
                    ;
B900                START:  EQU  0B900H
C000                GRAMS:  EQU  0C000H
6F6B                COLOR:  EQU  6F6BH
428B                CLS:    EQU  428BH                ;カーソル ケス
E626                MON:    EQU  0E626H
005C                GRAM0:  EQU  5CH
005D                GRAM1:  EQU  5DH
```

```
005E              GRAM2: EQU   5EH
005F              MRAM:  EQU   5FH
                  ;
                         ORG   START
                  ;
B900 326EBB              LD    (SPSAVE),A        ;ｹﾞﾝｻﾞｲ ﾉ ｽﾀｯｸ ｦ ﾀｲﾋ
B903 31ACBB              LD    SP,STACK          ;ｱﾗﾀﾆ ｽﾀｯｸ ｦ ｾｯﾃｲ
                  ;
B906 011950              LD    BC,5019H
B909 3E01                LD    A,01H
B90B 32B2E6              LD    (0E6B2H),A
B90E 3E19                LD    A,19H
B910 32B3E6              LD    (0E6B3H),A
B913 3EE8                LD    A,0E8H
B915 32B4E6              LD    (0E6B4H),A
B918 3E00                LD    A,00H
B91A 32B8E6              LD    (0E6B8H),A
B91D 3EFF                LD    A,0FFH
B91F 32B9E6              LD    (0E6B9H),A
B922 CD6B6F              CALL  COLOR
                  ;
B925 CD8B42              CALL  CLS               ;ｶｰｿﾙ OFF
                  ;
B928 2100C0       MAIN:  LD    HL,GRAMS
B92B 110000              LD    DE,0000H          ;ﾊｼﾞﾒﾉ ﾋｮｳｼﾞｲﾁ
B92E 3E27                LD    A,27H
B930 23           ML1:   INC   HL
B931 14                  INC   D
B932 3D                  DEC   A
B933 C230B9              JP    NZ,ML1
B936 3E64                LD    A,100
B938 015000              LD    BC,0050H
B93B 09           ML2:   ADD   HL,BC
B93C 1C                  INC   E
B93D 3D                  DEC   A
B93E C23BB9              JP    NZ,ML2
                  ;
B941 0603                LD    B,03H
                  ;
B943 78           KAITEN:LD    A,B
B944 FE03                CP    03H
B946 C257B9              JP    NZ,CD2
B949 E5                  PUSH  HL
B94A D5                  PUSH  DE
B94B 112ABA              LD    DE,CDATA1
B94E CD0BBA              CALL  DISP
B951 05                  DEC   B
B952 D1                  POP   DE
B953 E1                  POP   HL
B954 C384B9              JP    WAIT
                  ;
B957 78           CD2:   LD    A,B
B958 FE02                CP    02H
B95A C26BB9              JP    NZ,CD3
B95D E5                  PUSH  HL
B95E D5                  PUSH  DE
B95F 1196BA              LD    DE,CDATA2
B962 CD0BBA              CALL  DISP
B965 05                  DEC   B
B966 D1                  POP   DE
```

```
B967 E1              POP  HL
B968 C384B9          JP   WAIT
                ;
B96B E5         CD3:     PUSH HL
B96C D5                  PUSH DE
B96D 1102BB              LD   DE,CDATA3
B970 CD0BBA              CALL DISP
B973 CDFCB9              CALL WAT
B976 D1                  POP  DE
B977 E1                  POP  HL
B978 E5                  PUSH HL
B979 D5                  PUSH DE
B97A 1196BA              LD   DE,CDATA2
B97D CD0BBA              CALL DISP
B980 0603                LD   B,03H
B982 D1                  POP  DE
B983 E1                  POP  HL
                ;
B984 CDFCB9     WAIT:    CALL WAT
                ;
B987 C398B9              JP   KIY
                ;
B98A DB09       WKIY:    IN   A,(09H)
B98C E601                AND  01H
B98E C243B9              JP   NZ,KAITEN
                ;
B991 ED7B6EBB            LD   SP,(SPSAVE)          ;モト スタック ニ モドス
                ;
B995 C326E6              JP   MON
                ;
B998 C5         KIY:     PUSH BC
B999 DB00                IN   A,(00H)
B99B 47                  LD   B,A
B99C E604                AND  04H
B99E CAB8B9              JP   Z,DOWN
                ;
B9A1 78                  LD   A,B
B9A2 E610                AND  10H
B9A4 CAC8B9              JP   Z,LEFT
                ;
B9A7 78                  LD   A,B
B9A8 E640                AND  40H
B9AA CAD9B9              JP   Z,RIGHT
                ;
B9AD DB01                IN   A,(01H)
B9AF E601                AND  01H
B9B1 CAEAB9              JP   Z,UP
                ;
B9B4 C1         KIYPOP:POP    BC
                ;
B9B5 C38AB9              JP   WKIY
                ;
B9B8 7B         DOWN:    LD   A,E
B9B9 FEB5                CP   0B5H
B9BB CAB4B9              JP   Z,KIYPOP
B9BE C5                  PUSH BC
B9BF 015000              LD   BC,0050H
B9C2 09                  ADD  HL,BC
B9C3 C1                  POP  BC
B9C4 1C                  INC  E
```

```
B9C5 C3B4B9            JP    KIYPOP
                ;
B9C8 7A         LEFT:  LD    A,D
B9C9 FE00              CP    00H
B9CB CAB4B9            JP    Z,KIYPOP
B9CE 2B                DEC   HL
B9CF 15                DEC   D
B9D0 CDFCB9            CALL  WAT
B9D3 CDFCB9            CALL  WAT
B9D6 C3B4B9            JP    KIYPOP
                ;
B9D9 7A         RIGHT: LD    A,D
B9DA FE4A              CP    4AH
B9DC CAB4B9            JP    Z,KIYPOP
B9DF 23                INC   HL
B9E0 14                INC   D
B9E1 CDFCB9            CALL  WAT
B9E4 CDFCB9            CALL  WAT
B9E7 C3B4B9            JP    KIYPOP
                ;
B9EA 7B         UP:    LD    A,E
B9EB FE00              CP    00H
B9ED CAB4B9            JP    Z,KIYPOP
B9F0 C5                PUSH  BC
B9F1 015000            LD    BC,0050H
B9F4 A7                AND   A
B9F5 ED42              SBC   HL,BC
B9F7 C1                POP   BC
B9F8 1D                DEC   E
B9F9 C3B4B9            JP    KIYPOP


B9FC D5         WAT:   PUSH  DE              ;ジカン マチ
B9FD 1607              LD    D,07H
B9FF 1EEF       W2:    LD    E,0EFH
BA01 1D         W1:    DEC   E
BA02 C201BA            JP    NZ,W1
BA05 15                DEC   D
BA06 C2FFB9            JP    NZ,W2
BA09 D1                POP   DE
BA0A C9                RET
                ;
                ;
BA0B F3         DISP:  DI                    ;グラフ ノ ヒョウジ ルーチン ワリコミ NO
BA0C D35E              OUT   (GRAM2),A       ;カラー ノ イロ グリーン
BA0E C5                PUSH  BC
BA0F 0E12              LD    C,12H
BA11 0606       LOP2:  LD    B,06H
BA13 1A         LOP1:  LD    A,(DE)
BA14 77                LD    (HL),A
BA15 23                INC   HL
BA16 13                INC   DE
BA17 05                DEC   B
BA18 C213BA            JP    NZ,LOP1
BA1B C5                PUSH  BC
BA1C 014A00            LD    BC,004AH
BA1F 09                ADD   HL,BC
BA20 C1                POP   BC
BA21 0D                DEC   C
BA22 C211BA            JP    NZ,LOP2
BA25 C1                POP   BC
```

```
BA26 D35F               OUT    (MRAM),A                  ;メイン メモリー ニ スル
BA28 FB                 EI                               ;ワリコミ OK
                  :
BA29 C9                 RET
                  :
                  :
                  :
BA2A 00000000  CDATA1:DB    00H,00H,00H,00H,00H,00H
BA2E 0000
BA30 00001800           DB    00H,00H,18H,00H,00H,00H
BA34 0000
BA36 00FFFFFF           DB    00H,0FFH,0FFH,0FFH,00H,00H
BA3A 0000
BA3C 00FFFFFF           DB    00H,0FFH,0FFH,0FFH,07H,00H
BA40 0700
BA42 00001800           DB    00H,00H,18H,00H,0FH,00H
BA46 0F00
BA48 0001FFC0           DB    00H,01H,0FFH,0C0H,1FH,00H
BA4C 1F00
BA4E 00031FE0           DB    00H,03H,1FH,0E0H,3FH,00H
BA52 3F00
BA54 000C1FFF           DB    00H,0CH,1FH,0FFH,0FFH,00H
BA58 FF00
BA5A 00301FFF           DB    00H,30H,1FH,0FFH,0FFH,00H
BA5E FF00
BA60 00C01FFF           DB    00H,0C0H,1FH,0FFH,0FEH,00H
BA64 FE00
BA66 00FFFFE0           DB    00H,0FFH,0FFH,0E0H,00H,00H
BA6A 0000
BA6C 00FFFFC0           DB    00H,0FFH,0FFH,0C0H,00H,00H
BA70 0000
BA72 00FFFF80           DB    00H,0FFH,0FFH,80H,00H,00H
BA76 0000
BA78 003FFC00           DB    00H,3FH,0FCH,00H,00H,00H
BA7C 0000
BA7E 00030C00           DB    00H,03H,0CH,00H,00H,00H
BA82 0000
BA84 00C30C00           DB    00H,0C3H,0CH,00H,00H,00H
BA88 0000
BA8A 007FFFE0           DB    00H,7FH,0FFH,0E0H,00H,00H
BA8E 0000
BA90 00000000           DB    00H,00H,00H,00H,00H,00H
BA94 0000
                  :
                  :
                  :
BA96 00000000  CDATA2:DB    00H,00H,00H,00H,00H,00H
BA9A 0000
BA9C 00001800           DB    00H,00H,18H,00H,00H,00H
BAA0 0000
BAA2 0000FF00           DB    00H,00H,0FFH,00H,00H,00H
BAA6 0000
BAA8 0000FF00           DB    00H,00H,0FFH,00H,07H,00H
BAAC 0700
BAAE 00001800           DB    00H,00H,18H,00H,0FH,00H
BAB2 0F00
BAB4 0001FFC0           DB    00H,01H,0FFH,0C0H,1FH,00H
BAB8 1F00
BABA 00031FE0           DB    00H,03H,1FH,0E0H,3FH,00H
BABE 3F00
BAC0 000C1FFF           DB    00H,0CH,1FH,0FFH,0FFH,00H
```

```
BAC4 FF00
BAC6 00301FFF            DB     00H,30H,1FH,0FFH,0FFH,00H
BACA FF00
BACC 00C01FFF            DB     00H,0C0H,1FH,0FFH,0FEH,00H
BAD0 FE00
BAD2 00FFFFE0            DB     00H,0FFH,0FFH,0E0H,00H,00H
BAD6 0000
BAD8 00FFFFC0            DB     00H,0FFH,0FFH,0C0H,00H,00H
BADC 0000
BADE 00FFFF80            DB     00H,0FFH,0FFH,80H,00H,00H
BAE2 0000
BAE4 003FFC00            DB     00H,3FH,0FCH,00H,00H,00H
BAE8 0000
BAEA 00030C00            DB     00H,03H,0CH,00H,00H,00H
BAEE 0000
BAF0 00C30C00            DB     00H,0C3H,0CH,00H,00H,00H
BAF4 0000
BAF6 007FFFE0            DB     00H,7FH,0FFH,0E0H,00H,00H
BAFA 0000
BAFC 00000000            DB     00H,00H,00H,00H,00H,00H
BB00 0000
                  ;
BB02 00000000 CDATA3:DB     00H,00H,00H,00H,00H,00H
BB06 0000
BB08 00001800            DB     00H,00H,18H,00H,00H,00H
BB0C 0000
BB0E 00001800            DB     00H,00H,18H,00H,00H,00H
BB12 0000
BB14 00001800            DB     00H,00H,18H,00H,07H,00H
BB18 0700
BB1A 00001800            DB     00H,00H,18H,00H,0FH,00H
BB1E 0F00
BB20 0001FFC0            DB     00H,01H,0FFH,0C0H,1FH,00H
BB24 1F00
BB26 00031FE0            DB     00H,03H,1FH,0E0H,3FH,00H
BB2A 3F00
BB2C 000C1FFF            DB     00H,0CH,1FH,0FFH,0FFH,00H
BB30 FF00
BB32 00301FFF            DB     00H,30H,1FH,0FFH,0FFH,00H
BB36 FF00
BB38 00C01FFF            DB     00H,0C0H,1FH,0FFH,0FEH,00H
BB3C FE00
BB3E 00FFFFE0            DB     00H,0FFH,0FFH,0E0H,00H,00H
BB42 0000
BB44 00FFFFC0            DB     00H,0FFH,0FFH,0C0H,00H,00H
BB48 0000
BB4A 00FFFF80            DB     00H,0FFH,0FFH,80H,00H,00H
BB4E 0000
BB50 003FFC00            DB     00H,3FH,0FCH,00H,00H,00H
BB54 0000
BB56 00030C00            DB     00H,03H,0CH,00H,00H,00H
BB5A 0000
BB5C 00C30C00            DB     00H,0C3H,0CH,00H,00H,00H
BB60 0000
BB62 007FFFE0            DB     00H,7FH,0FFH,0E0H,00H,00H
BB66 0000
BB68 00000000            DB     00H,00H,00H,00H,00H,00H
BB6C 0000
                  ;
                  ;
```

```
BB6E                SPSAVE:DS    02H
BB70                       DS    60
BBAC                STACK:
                    ;
BBAC                       END
```

## ■ヘリコプターを８方向に移動させる

また，８方向に移動させるプログラムも**リスト4-5**に示します．

これで第４章を終りますが，グラフィックスの原理や表示のやり方が理解できたでしょうか．次章は横方向もドット単位で移動させますので，動きがもっとスムーズになります．

**●リスト4-5 ヘリコプターの移動(8方向)**

```
                    ;*****************************************************
                    ;
                    ;GRPH--5 ﾍﾘｺﾌﾟﾀｰ ﾉ ﾊﾈ ﾉ ｶｲﾃﾝ ﾄ ｷｰﾆｭｳﾘｮｸ 8ﾎｳｺｳ
                    ;
                    ;PROGRAM NAME ---> Grph5.e
                    ;
                    ;*****************************************************
                    ;
B900                START: EQU   0B900H
C000                GRAMS: EQU   0C000H
6F6B                COLOR: EQU   6F6BH
428B                CLS:   EQU   428BH           ;ｶｰｿﾙ ｹｽ
E626                MON:   EQU   0E626H
005C                GRAM0: EQU   5CH
005D                GRAM1: EQU   5DH
005E                GRAM2: EQU   5EH
005F                MRAM:  EQU   5FH
                    ;
                           ORG   START
                    ;
B900 ED73E8BB              LD    (SPSAVE),SP     ;ｽﾀｯｸ ﾉ ﾀｲﾋ
B904 3126BC                LD    SP,STACK
                    ;
B907 011950                LD    BC,5019H
B90A 3E01                  LD    A,01H
B90C 32B2E6                LD    (0E6B2H),A
B90F 3E19                  LD    A,19H
B911 32B3E6                LD    (0E6B3H),A
B914 3EE8                  LD    A,0E8H
B916 32B4E6                LD    (0E6B4H),A
B919 3E00                  LD    A,00H
B91B 32B8E6                LD    (0E6B8H),A
B91E 3EFF                  LD    A,0FFH
B920 32B9E6                LD    (0E6B9H),A
B923 CD6B6F                CALL  COLOR
                    ;
B926 CD8B42                CALL  CLS
                    ;
B929 2100C0         MAIN:  LD    HL,GRAMS
B92C 110000                LD    DE,0000H        ;ﾊｼﾞﾒﾉ ﾋｮｳｼﾞｲﾁ
```

```
B92F 3E27              LD    A,27H
B931 23       ML1:     INC   HL
B932 14                INC   D
B933 3D                DEC   A
B934 C231B9            JP    NZ,ML1
B937 3E64              LD    A,100
B939 015000            LD    BC,0050H
B93C 09       ML2:     ADD   HL,BC
B93D 1C                INC   E
B93E 3D                DEC   A
B93F C23CB9            JP    NZ,ML2
              ;
B942 0603              LD    B,03H
              ;
B944 78       KAITEN:LD      A,B
B945 FE03              CP    03H
B947 C258B9            JP    NZ,CD2
B94A E5                PUSH  HL
B94B D5                PUSH  DE
B94C 11A4BA            LD    DE,CDATA1
B94F CD85BA            CALL  DISP
B952 05                DEC   B
B953 D1                POP   DE
B954 E1                POP   HL
B955 C385B9            JP    WAIT
              ;
B958 78       CD2:     LD    A,B
B959 FE02              CP    02H
B95B C26CB9            JP    NZ,CD3
B95E E5                PUSH  HL
B95F D5                PUSH  DE
B960 1110BB            LD    DE,CDATA2
B963 CD85BA            CALL  DISP
B966 05                DEC   B
B967 D1                POP   DE
B968 E1                POP   HL
B969 C385B9            JP    WAIT
              ;
B96C E5       CD3:     PUSH  HL
B96D D5                PUSH  DE
B96E 117CBB            LD    DE,CDATA3
B971 CD85BA            CALL  DISP
B974 CD76BA            CALL  WAT
B977 D1                POP   DE
B978 E1                POP   HL
B979 E5                PUSH  HL
B97A D5                PUSH  DE
B97B 1110BB            LD    DE,CDATA2
B97E CD85BA            CALL  DISP
B981 0603              LD    B,03H
B983 D1                POP   DE
B984 E1                POP   HL
              ;
B985 CD76BA   WAIT:    CALL  WAT
              ;
B988 C399B9            JP    KIY
              ;
B98B DB09     WKIY:    IN    A,(09H)
B98D E601              AND   01H
B98F C244B9            JP    NZ,KAITEN
```

```
                    ;
B992 ED7BE8BB            LD    SP,(SPSAVE)
                    ;
B996 C326E6              JP    MON
                    ;
B999 C5             KIY: PUSH  BC
B99A DB00                IN    A,(00H)        ;1
B99C 47                  LD    B,A
B99D E602                AND   02H
B99F CAD2B9              JP    Z,LDOWN
                    ;
B9A2 78                  LD    A,B            ;2
B9A3 E604                AND   04H
B9A5 CAE9B9              JP    Z,DOWN
                    ;
B9A8 78                  LD    A,B            ;3
B9A9 E608                AND   08H
B9AB CAF9B9              JP    Z,RDOWN
                    ;
                    ;
B9AE 78                  LD    A,B            ;4
B9AF E610                AND   10H
B9B1 CA10BA              JP    Z,LEFT
                    ;
B9B4 78                  LD    A,B            ;6
B9B5 E640                AND   40H
B9B7 CA21BA              JP    Z,RIGHT
                    ;
B9BA 78                  LD    A,B            ;7
B9BB E680                AND   80H
B9BD CA32BA              JP    Z,LUP
                    ;


B9C0 DB01                IN    A,(01H)        ;8
B9C2 47                  LD    B,A
B9C3 E601                AND   01H
B9C5 CA4BBA              JP    Z,UP
                    ;
B9C8 78                  LD    A,B            ;9
B9C9 E602                AND   02H
B9CB CA5DBA              JP    Z,RUP
                    ;
B9CE C1             KIYPOP:POP BC
                    ;
B9CF C38BB9              JP    WKIY
                    ;
B9D2 7B             LDOWN: LD  A,E
B9D3 FEB5                CP    0B5H
B9D5 CACEB9              JP    Z,KIYPOP
B9D8 7A                  LD    A,D
B9D9 FE00                CP    00H
B9DB CACEB9              JP    Z,KIYPOP
B9DE C5                  PUSH  BC
B9DF 014F00              LD    BC,004FH
B9E2 09                  ADD   HL,BC
B9E3 C1                  POP   BC
B9E4 1C                  INC   E
B9E5 15                  DEC   D
B9E6 C3CEB9              JP    KIYPOP
```

```
B9E9 7B       DOWN:  LD   A,E
B9EA FEB5            CP   0B5H
B9EC CACEB9          JP   Z,KIYPOP
B9EF C5              PUSH BC
B9F0 015000          LD   BC,0050H
B9F3 09              ADD  HL,BC
B9F4 C1              POP  BC
B9F5 1C              INC  E
B9F6 C3CEB9          JP   KIYPOP
                ;
B9F9 7B       RDOWN: LD   A,E
B9FA FEB5            CP   0B5H
B9FC CACEB9          JP   Z,KIYPOP
B9FF 7A              LD   A,D
BA00 FE4A            CP   4AH
BA02 CACEB9          JP   Z,KIYPOP
BA05 C5              PUSH BC
BA06 015100          LD   BC,0051H
BA09 09              ADD  HL,BC
BA0A C1              POP  BC
BA0B 1C              INC  E
BA0C 14              INC  D
BA0D C3CEB9          JP   KIYPOP
                ;
BA10 7A       LEFT:  LD   A,D
BA11 FE00            CP   00H
BA13 CACEB9          JP   Z,KIYPOP
BA16 2B              DEC  HL
BA17 15              DEC  D
BA18 CD76BA          CALL WAT
BA1B CD76BA          CALL WAT
BA1E C3CEB9          JP   KIYPOP
                ;
BA21 7A       RIGHT: LD   A,D
BA22 FE4A            CP   4AH
BA24 CACEB9          JP   Z,KIYPOP
BA27 23              INC  HL
BA28 14              INC  D
BA29 CD76BA          CALL WAT
BA2C CD76BA          CALL WAT
BA2F C3CEB9          JP   KIYPOP
                ;
BA32 7B       LUP:   LD   A,E
BA33 FE00            CP   00H
BA35 CACEB9          JP   Z,KIYPOP
BA38 7A              LD   A,D
BA39 FE00            CP   00H
BA3B CACEB9          JP   Z,KIYPOP
BA3E C5              PUSH BC
BA3F 015100          LD   BC,51H
BA42 A7              AND  A
BA43 ED42            SBC  HL,BC
BA45 C1              POP  BC
BA46 1D              DEC  E
BA47 15              DEC  D
BA48 C3CEB9          JP   KIYPOP
                ;
BA4B 7B       UP:    LD   A,E
BA4C FE00            CP   00H
BA4E CACEB9          JP   Z,KIYPOP
BA51 C5              PUSH BC
```

```
BA52 015000               LD    BC,0050H
BA55 A7                   AND   A
BA56 ED42                 SBC   HL,BC
BA58 C1                   POP   BC
BA59 1D                   DEC   E
BA5A C3CEB9               JP    KIYPOP
                 ;
BA5D 7B          RUP:     LD    A,E
BA5E FE00                 CP    00H
BA60 CACEB9               JP    Z,KIYPOP
BA63 7A                   LD    A,D
BA64 FE4A                 CP    4AH
BA66 CACEB9               JP    Z,KIYPOP
BA69 C5                   PUSH  BC
BA6A 014F00               LD    BC,004FH
BA6D A7                   AND   A
BA6E ED42                 SBC   HL,BC
BA70 C1                   POP   BC
BA71 1D                   DEC   E
BA72 14                   INC   D
BA73 C3CEB9               JP    KIYPOP
                 ;
                 ;
BA76 D5          WAT:     PUSH  DE                          ;ジ゛カン マチ
BA77 1604                 LD    D,04H
BA79 1EEF        W2:      LD    E,0EFH
BA7B 1D          W1:      DEC   E
BA7C C27BBA               JP    NZ,W1
BA7F 15                   DEC   D
BA80 C279BA               JP    NZ,W2
BA83 D1                   POP   DE
BA84 C9                   RET
                 ;
                 ;
BA85 F3          DISP:    DI                                ;ク゛ラフ ノ ヒョウシ゛ ルーチン ワリコミ NO
BA86 D35E                 OUT   (GRAM2),A                   ;カラー ノ イロ ク゛リーン
BA88 C5                   PUSH  BC
BA89 0E12                 LD    C,12H
BA8B 0606        LOP2:    LD    B,06H
BA8D 1A          LOP1:    LD    A,(DE)
BA8E 77                   LD    (HL),A
BA8F 23                   INC   HL
BA90 13                   INC   DE
BA91 05                   DEC   B
BA92 C28DBA               JP    NZ,LOP1
BA95 C5                   PUSH  BC
BA96 014A00               LD    BC,004AH
BA99 09                   ADD   HL,BC
BA9A C1                   POP   BC
BA9B 0D                   DEC   C
BA9C C28BBA               JP    NZ,LOP2
BA9F C1                   POP   BC
BAA0 D35F                 OUT   (MRAM),A                    ;メイン メモリー ニ スル
BAA2 FB                   EI                                ;ワリコミ OK
                 ;
BAA3 C9                   RET
                 ;
                 ;
                 ;
BAA4 00000000    CDATA1:DB      00H,00H,00H,00H,00H,00H
BAA8 0000
```

```
BAAA 00001800          DB    00H,00H,18H,00H,00H,00H
BAAE 0000
BAB0 00FFFFFF          DB    00H,0FFH,0FFH,0FFH,00H,00H
BAB4 0000
BAB6 00FFFFFF          DB    00H,0FFH,0FFH,0FFH,07H,00H
BABA 0700
BABC 00001800          DB    00H,00H,18H,00H,0FH,00H
BAC0 0F00
BAC2 0001FFC0          DB    00H,01H,0FFH,0C0H,1FH,00H
BAC6 1F00
BAC8 00031FE0          DB    00H,03H,1FH,0E0H,3FH,00H
BACC 3F00
BACE 000C1FFF          DB    00H,0CH,1FH,0FFH,0FFH,00H
BAD2 FF00
BAD4 00301FFF          DB    00H,30H,1FH,0FFH,0FFH,00H
BAD8 FF00
BADA 00C01FFF          DB    00H,0C0H,1FH,0FFH,0FEH,00H
BADE FE00
BAE0 00FFFFE0          DB    00H,0FFH,0FFH,0E0H,00H,00H
BAE4 0000
BAE6 00FFFFC0          DB    00H,0FFH,0FFH,0C0H,00H,00H
BAEA 0000
BAEC 00FFFF80          DB    00H,0FFH,0FFH,80H,00H,00H
BAF0 0000
BAF2 003FFC00          DB    00H,3FH,0FCH,00H,00H,00H
BAF6 0000
BAF8 00030C00          DB    00H,03H,0CH,00H,00H,00H
BAFC 0000
BAFE 00C30C00          DB    00H,0C3H,0CH,00H,00H,00H
BB02 0000
BB04 007FFFE0          DB    00H,7FH,0FFH,0E0H,00H,00H
BB08 0000
BB0A 00000000          DB    00H,00H,00H,00H,00H,00H
BB0E 0000
                  :
                  :
                  :
BB10 00000000 CDATA2:DB    00H,00H,00H,00H,00H,00H
BB14 0000
BB16 00001800          DB    00H,00H,18H,00H,00H,00H
BB1A 0000
BB1C 0000FF00          DB    00H,00H,0FFH,00H,00H,00H
BB20 0000
BB22 0000FF00          DB    00H,00H,0FFH,00H,07H,00H
BB26 0700
BB28 00001800          DB    00H,00H,18H,00H,0FH,00H
BB2C 0F00
BB2E 0001FFC0          DB    00H,01H,0FFH,0C0H,1FH,00H
BB32 1F00
BB34 00031FE0          DB    00H,03H,1FH,0E0H,3FH,00H
BB38 3F00
BB3A 000C1FFF          DB    00H,0CH,1FH,0FFH,0FFH,00H
BB3E FF00
BB40 00301FFF          DB    00H,30H,1FH,0FFH,0FFH,00H
BB44 FF00
BB46 00C01FFF          DB    00H,0C0H,1FH,0FFH,0FEH,00H
BB4A FE00
BB4C 00FFFFE0          DB    00H,0FFH,0FFH,0E0H,00H,00H
BB50 0000
BB52 00FFFFC0          DB    00H,0FFH,0FFH,0C0H,00H,00H
```

```
BB56 0000
BB58 00FFFF80            DB    00H,0FFH,0FFH,80H,00H,00H
BB5C 0000
BB5E 003FFC00            DB    00H,3FH,0FCH,00H,00H,00H
BB62 0000
BB64 00030C00            DB    00H,03H,0CH,00H,00H,00H
BB68 0000
BB6A 00C30C00            DB    00H,0C3H,0CH,00H,.00H,00H
BB6E 0000
BB70 007FFFE0            DB    00H,7FH,0FFH,0E0H,00H,00H
BB74 0000
BB76 00000000            DB    00H,00H,00H,00H,00H,00H
BB7A 0000
                 ;
BB7C 00000000    CDATA3:DB     00H,00H,00H,00H,00H,00H
BB80 0000
BB82 00001800            DB    00H,00H,18H,00H,00H,00H
BB86 0000
BB88 00001800            DB    00H,00H,18H,00H,00H,00H
BB8C 0000
BB8E 00001800            DB    00H,00H,18H,00H,07H,00H
BB92 0700
BB94 00001800            DB    00H,00H,18H,00H,0FH,00H
BB98 0F00
BB9A 0001FFC0            DB    00H,01H,0FFH,0C0H,1FH,00H
BB9E 1F00
BBA0 00031FE0            DB    00H,03H,1FH,0E0H,3FH,00H
BBA4 3F00
BBA6 000C1FFF            DB    00H,0CH,1FH,0FFH,0FFH,00H
BBAA FF00
BBAC 00301FFF            DB    00H,30H,1FH,0FFH,0FFH,00H
BBB0 FF00
BBB2 00C01FFF            DB    00H,0C0H,1FH,0FFH,0FEH,00H
BBB6 FE00
BBB8 00FFFFE0            DB    00H,0FFH,0FFH,0E0H,00H,00H
BBBC 0000
BBBE 00FFFFC0            DB    00H,0FFH,0FFH,0C0H,00H,00H
BBC2 0000
BBC4 00FFFF80            DB    00H,0FFH,0FFH,80H,00H,00H
BBC8 0000
BBCA 003FFC00            DB    00H,3FH,0FCH,00H,00H,00H
BBCE 0000
BBD0 00030C00            DB    00H,03H,0CH,00H,00H,00H
BBD4 0000
BBD6 00C30C00            DB    00H,0C3H,0CH,00H,00H,00H
BBDA 0000
BBDC 007FFFE0            DB    00H,7FH,0FFH,0E0H,00H,00H
BBE0 0000
BBE2 00000000            DB    00H,00H,00H,00H,00H,00H
BBE6 0000
                 ;
                 ;
BBE8             SPSAVE:DS     02H
BBEA                    DS     60
BC26             STACK:
                 ;
BC26                    END
```

# 第5章
# スムーズ・スクロール

横方向をドット単位で動かす

この章では，先に使ったヘリコプターをさらにスムーズに移動させる方法を考えます．つまり，横方向もドット単位で移動させせようというわけです．

しかし，VRAMの関係上，8ビットを1単位として扱わなければならないので，横の動きは縦ほど簡単にはいきません．まず，横方向の移動に必要なビットのシフト命令から学んで，スムーズな動きのプログラムを作ります．

また，テキスト画面とグラフィックス画面を重ね合わせたり，動きを速くするためにDMAを止めたりといったテクニックも紹介しています．

*Section 1*

# スムーズな動きの原理

Smooth Scroll

## 右へシフトするには問題がある

ヘリコプターをスムーズに動かすためには，縦方向だけでなく横方向もドット単位で動くようにしなければなりません．そこで，ヘリコプターのパターン・データを表示する前に，左右へビット単位でシフト（桁送り）させます．シフトは簡単なようで結構手間がかかります．まず，右へのシフトを考えてみましょう（**図5-1**）．

**図5-1　右へのシフト**

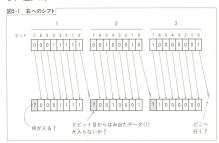



**図5-1**からわかるように，右へシフトする場合には２つ問題があります．

1．７ビット目には何が入るか？

2．０ビット目からはみ出したデータは消えてしまうのか？

この２つの問題を解決しなければなりません．そのためには，右へシフトする

ときにデータが次のように動いてくれれば良いわけです．

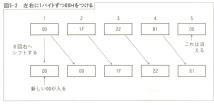
1．1バイト目のデータを右へ1ビットシフトしたときには，7ビット目に0が入る．
2．1バイト目のデータを右へ1ビットシフトしたときには，0ビット目のデータが2バイト目のデータの7ビット目に入る．
3．2バイト目のデータの場合は，7ビット目には前のデータ（1バイト目0ビット目のデータ）が入り，0ビット目のデータは3バイト目のデータの7ビット目に入る．
4．3バイト目のデータも，2バイト目のデータと同じシフトを行う．

しかし，この4つ全部を満たしたとしても，まだ問題があります．8回右へシフトしたら，データは1バイト目と3バイト目のデータが00Hになってしまいます．困りました．ここで，ヘリコプターのパターンを表示したときのことを思い出してみましょう．

ヘリコプターのデータは1列が6バイトのデータの集りでした．そのうちの1バイト目と6バイト目は，すべて00Hにしてありました．この00Hは，8ビット左右へ移動したときにデータを保存するという役目を持っていたのです．ですから，左右に1バイトずつ00Hのデータをつけておけば，8回までビットをシフトすることができます(**図5-2**)．8回を越えた場合はVRAMの番地を1番地分進めればよいので，8ビット分のシフトさえできれば大丈夫です．また，左へのシフトも右へのシフトと同じ原理なのでわかると思います．

## ■はみ出たデータはどこへ行く

0ビット目のデータを右へシフトすると，はみ出たデータはどうなるのでし

**図5-2　左右に1バイトずつ00Hをつける**

ょうか．このはみ出たデータを保存するフラグがあります．それがCy(キャリー）です．Cy フラグはF（フラグ）レジスタのひとつです．

## ■仮想CY(キャリー)レジスタとCy(キャリー)フラグの考え方

CPU の内部にあるF（フラグ）レジスタのフラグは全部で6個ありました．本来，フラグはブランチ（Branch；分岐）機能のためにあり，条件判断に多く使用されています．ここでは，フラグの状態が1か0かによって条件を判断しているだけで，各フラグに1を加えたり引いたりといった演算処理はしていません．

しかし，Cy フラグだけは他のフラグと違って，強制的に1をセットされたり0をセットされたりします．また，桁あふれ（オーバーフロー）を起こしたビットのデータを蓄えたり，蓄えたデータを次の演算で汎用レジスタに加えたりもします．条件判断の目印としてよりは，汎用レジスタから桁あふれを起こした1ビットを蓄える場所として使用される方が多いくらいです．したがって，Cy フラグは，本来の機能とレジスタに近い機能の両方を持っていると言うことができます．

そこで本書では，Cy フラグの複雑な機能をわかりやすくするために，CY レジスタという仮想のレジスタを考えました．1ビットデータの蓄えや，蓄えたデータが利用される場合はCY レジスタとして，本来の機能である条件判断のために利用される場合はCy フラグとして扱っており，同じキャリー・フラグを使い方によって2つに分けています．たとえば，Aレジスタを右へ1ビットシフト（動かす）する場合を考えると，**図5-3**のようになります．

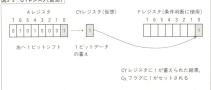
**図5-3 CYレジスタ(仮想)**

では，CY レジスタにデータが保存されるようすを**図5-4**で説明しましょう．

**図5-4　CYレジスタにデータが保存されるようす**

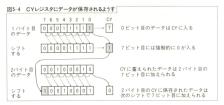



**図5-4**で使用されているシフト命令は，次の4つです（**図5-5**）．

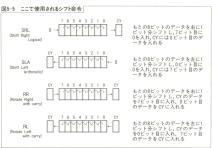
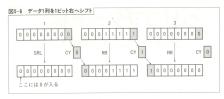
**図5-5　ここで使用されるシフト命令**

この4つのシフト命令を使用すれば，ヘリコプターをスムーズに動かすことができます．これらを使って，ヘリコプターのデータ1列を1ビット右へシフトしましょう（**図5-6**）．これで全体が1ビット右へシフトしたことになります．

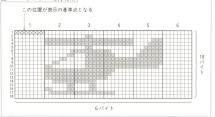
図5-6 データ1列を1ビット右へシフト

*Section 2*

# ヘリコプターを横にスムーズに動かす

Smooth Scroll

## ヘリコプターを表示させるには

横方向へのスムーズな動きの原理は，シフト命令を利用してドット単位でデータを移動させます．しかし，VRAM の関係上，シフト命令だけでは済みません．そこで，もう一度ヘリコプターの表示方法について説明しておくことにします．**図5-7**を見てください．

**図5-7　ヘリコプターの表示方法**

このヘリコプターのキャラクタは横６バイト，縦18バイトで構成されています．ヘリコプターを表示するために，HL ペアレジスタを VRAM のポインタ（アドレスの管理）として使用します．HL ペアレジスタは常に基準点の番地だけを管理しており，この基準点をもとに DISP というサブルーチンがヘリコプターのデータを DE ペアジスタで読み出してきて，横６バイト縦18バイトの計108バイトを表示します．

**■自由に移動できるのは，左右８ドットまで**

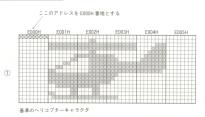
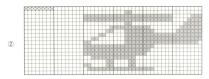
ここで基準点を変化させずに左右に移動できるのは，８ドットまでです．したがって，８ドットまでは自由にドット単位で移動できますが，それ以上はドット単位で移動できません．もし，そのまま移動してしまうと，１ドット単位でヘリコプターのキャラクタが消えていきます．

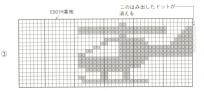
## 9ドット目をどう移動させるか

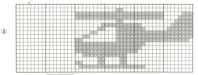
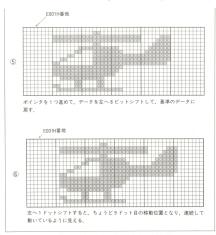
そこで，９ドット目の移動は少しやっかいになります．このままでは９ドット目の移動ができないので，ポインタ(HL ペアレジスタのデータ)を１つ進めて，次のアドレスから表示するようにします．しかし，そのままのデータでは左右どちらかに８ドットずれているので８ドット分ずれて表示することになり，スムーズに移動できません．そこで，８ドットずれたデータを一旦もとに戻してから，１ドットずらして表示することにします．こうすれば，問題なく移動することができます．

もう少し具体的な説明をしましょう．右へ移動する例を考えます．ヘリコプターのデータは，回転翼を回転させるために，３つのデータに分けてあります．この３つのデータを右へドット単位でシフトすれば，８ドットまでは問題なく移動できますが９ドット目からが問題です．図5-8で順を追ってみましょう．

図5-8　9ドット目からの移動

②

右へ8ドット移動したヘリコプターのキャラクタ．ここまでは問題なくドット単位で移動できる．

③

9ドット移動すると，右側の縦のドットが消えてしまう．
ポインタを1つ進めない限り，右へシフトするとシフトするたびにはみ出したドットが消えてしまう．

④

そこで②の状態のデータをそのままにして，ポインタ（VRAMのアドレス）を1つ進めると，バイト単位で移動するので1回で8ドット移動してしまう．

この図でわかるように，9ドット以上を移動させるためには

①→②→⑤→⑥→②→⑤→⑥→②

と繰り返せば良いのです．左へスムーズに移動させる方法も右と全く同じなので，特に説明しません．

## 途中で方向を変えて移動するとき

これで，右と左へスムーズに移動できるようになりましたが，まだ問題が残っています．右へ移動していて途中で左に移動するような場合はどうしたら良いのでしょうか．ヘリコプターのデータが基準に戻っていれば問題ないのです

が，**図5-9**のように途中だったらどうするのでしょうか．

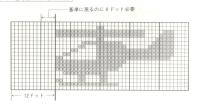
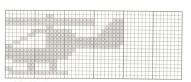
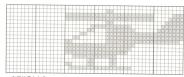
**図5-9　基準点に戻る途中のとき**

この位置から左端へ移動するドット数は12ドットとなります．移動する左右のドット数を正確にカウントしていないと，ヘリコプターの基準のデータに戻ることができません．そこで，左右へ移動したドット数をカウントするプログラムが必要で，そのデータを確保しておかなければなりません．**図5-10**を使って説明しましょう．

**図5-10　途中で方向を変える**

左端に来たとき
左へ移動するためにはポインタのデータから1を引いて，基準のデータまで戻してやる必要がある．
カウンタのデータ＝00(00H)
右端へ移動するのに，ポインタのデータを変えずに16ドット分移動できる．

基準のデータ……カウンタのデータ＝08

右端に来たとき
右へ移動するためには，ポインタのデータに1を加えて基準のデータまで戻してやる必要がある．
　　カウンタのデータ＝16(10H)
左端へ移動するのにポインタのデータを変えずに16ドット分移動できる．

## ■1ドットシフトするたびにカウンタのデータを調べればよい

**図5-10**を見ると，ポインタ（VRAMのアドレス）を次のアドレスにするか，ひとつ手前のアドレスにするかはカウンタのデータを参照すれば良いことがわかります．

つまり，カウンタのデータが0や16以外のときは，ポインタをいじらずに左右へデータをシフトすれば良いし，もし0であれば，データを8ドットシフトしてポインタを1つ減らせば良いことになります．したがって，1ドット左右どちらかにシフトするごとに，カウンタのデータを調べれば良いのです．

# スムーズな移動のプログラム

では，ヘリコプターを左右へスムーズに移動させるプログラムを**リスト5-1**に示します．プログラムは第4章の**リスト4-5**(P.185)と基本的には同じです．一部のキールーチン処理と横のシフトルーチンが加わっただけですので，左右のシフトルーチンを中心に説明していきます．

## ■キールーチンは少し簡単に

今までは左右への移動は，HLペアレジスタ(VRAMのアドレス，つまりポインタ)を±1することによって行いました．ですから，たとえば右へ移動する場合は，HLペアレジスタに1を加えてメイン・プログラムに戻れば良いだけでした．もちろん，右へ移動した回数をカウントしているDレジスタにも1を加えなければなりませんでした．

今回はその代りに，右へ1ドットずつ移動するルーチンを呼ぶので少し簡単になっています．ちょっと較べてみましょう．

今までのルーチン例

```
RIGHT：LD   A, D
       CP   4AH
       JP   Z, KIYPOP
       INC  HL
       INC  D
       JP   KIYPOP
```

今回のルーチン例

```
RIGHT：LD   A, D
       CP   4AH
       JP   Z, KIYPOP
       CALL RSM
       JP   KIYPOP
```

（右へ1ドットずつ移動するルーチンを呼ぶ）

見てわかるように今回の方が少し簡単になっています．RSMルーチンやLSMルーチンで，HLペアレジスタにいつ1を加えたり引いたりするのかを計算しています．

## ■左右へドット単位で移動するルーチンは

次に，左右へドット単位で移動するルーチンを説明しましょう．主に右へのドット移動のフローチャートを説明します．左へのドット移動のフローチャートは右とほとんど同じなので，大きく違う点だけを説明します．右へのドット移動のフローチャートが理解できれば，左へのフローチャートもすぐに理解できることと思います．

**図5-11　左へドット単位でデータを移動させるルーチン**

| 処理 | 説明 |
| --- | --- |
| RSM | |
| ((SP))←HL | ポインタ（VRAMのアドレス）をスタックに保存する |
| HL←SSCD | HLペアレジスタに，左右へ移動したドット数が保存してあるアドレス（SSCD）をロードする |
| A←(HL) | Aレジスタに移動したドット数を入れる |
| A−16 | 16ドット移動したか（右端まできたか） |
| A≠16（NO→RSM1，YES→下へ） | まだ右へ移動できるので，右へドット単位で移動するルーチンへ飛ぶ |
| ((SP))←DE | DEペアレジスタをスタックに保存 |
| D←08H | データを左へ8ドット分移動するための回数をセット（データを基本データに戻すため） |
| HL←CDATA1 | HLペアレジスタに，データ1のアドレスをセットする |
| LRSM | 左へ1ドット移動するサブルーチンを呼ぶ |
| HL←CDATA2 | HLペアレジスタに，データ2のアドレスをセットする |
| LRSM | 左へ1ドット移動するサブルーチンを呼ぶ |
| HL←CDATA3 | HLペアレジスタに，データ3のアドレスをセットする |
| LRSM | 左へ1ドット移動するサブルーチンを呼ぶ |
| D←D−1 | Dレジスタから1を引く |
| D≠0（YES→HL←CDATA1へ戻る，NO→下へ） | |
| DE←((SP)) | もとのデータに戻す |
| HL←((SP)) | もとのデータに戻す |
| D←D+1 | Dレジスタに1を加える（右へ1番地進んだことをカウントする） |

図5-12　3データを1ドット右へ移動するルーチン

図5-13　右へ1ドット移動するサブルーチン

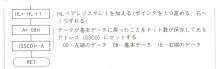
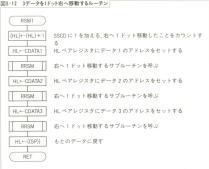
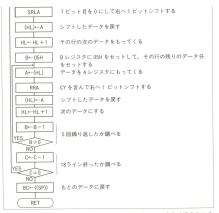
| 処理 | 説明 |
|---|---|
| SRLA | 7ビット目を0にして右へ1ビットシフトする |
| (HL)←A | シフトしたデータを戻す |
| HL←HL+1 | その行の次のデータをもってくる |
| B←05H | Bレジスタに05Hをセットして，その行の残りのデータ分をセットする |
| A←(HL) | データをAレジスタにもってくる |
| RRA | CYを含んで右へ1ビットシフトする |
| (HL)←A | シフトしたデータを戻す |
| HL←HL+1 | 次のデータにする |
| B←B−1<br>B≠0 (YES / NO) | 5回繰り返したか調べる |
| C←C−1<br>C≠0 (YES / NO) | 18ライン終ったか調べる |
| BC←((SP)) | もとのデータに戻す |
| RET | |

左へ移動するフローチャートは一部を除いて同じなので，大きく違うところだけを説明します．

**図5-14　左へ1ドット移動するルーチン**

**●リスト5-1 ヘリコプターのスムーズな移動**

```
               ;*********************************************************
               ;
               ;GRPH--8 ﾖｺ ﾉ ｽﾑｰｽﾞ ｲﾄﾞｳ  (ﾍﾘｺﾌﾟﾀｰ ﾉ ﾊﾈ ﾉ ｶｲﾃﾝ ﾄ ｷｰﾆｭｳﾘｮｸ  8ﾎｳｺｳ
               ;
               ;PROGRAM NAME ---> Grph8.e
               ;
               ;*********************************************************
               ;
B900           START:  EQU  0B900H
C000           GRAMS:  EQU  0C000H
6F6B           COLOR:  EQU  6F6BH
428B           CLS:    EQU  428BH              ;ｶｰｿﾙ ｹｽ
```

```
E626            MON:    EQU  0E626H
005C            GRAM0:  EQU  5CH
005D            GRAM1:  EQU  5DH
005E            GRAM2:  EQU  5EH
005F            MRAM:   EQU  5FH
                ;
                        ORG  START
                ;
B900 ED73A6BC           LD   (SPSAVE),SP        ;スタック ノ タイヒ
B904 31E4BC             LD   SP,STACK
                ;
B907 011950             LD   BC,5019H
B90A 3E01               LD   A,01H
B90C 32B2E6             LD   (0E6B2H),A
B90F 3E19               LD   A,19H
B911 32B3E6             LD   (0E6B3H),A
B914 3EE8               LD   A,0E8H
B916 32B4E6             LD   (0E6B4H),A
B919 3E00               LD   A,00H
B91B 32B8E6             LD   (0E6B8H),A
B91E 3EFF               LD   A,0FFH
B920 32B9E6             LD   (0E6B9H),A
B923 CD6B6F             CALL COLOR
                ;
B926 CD8B42             CALL CLS
                ;
B929 3E08       MAIN:   LD   A,08H
B92B 3261BB             LD   (SSCD),A
                ;
B92E 2100C0             LD   HL,GRAMS
B931 110000             LD   DE,0000H           ;ハジメノ ヒョウジイチ
B934 3E27               LD   A,27H
B936 23         ML1:    INC  HL
B937 14                 INC  D
B938 3D                 DEC  A
B939 C236B9             JP   NZ,ML1
B93C 3E64               LD   A,100
B93E 015000             LD   BC,0050H
B941 09         ML2:    ADD  HL,BC
B942 1C                 INC  E
B943 3D                 DEC  A
B944 C241B9             JP   NZ,ML2
                ;
B947 0603               LD   B,03H
                ;
B949 78         KAITEN:LD    A,B
B94A FE03               CP   03H
B94C C25DB9             JP   NZ,CD2
B94F E5                 PUSH HL
B950 D5                 PUSH DE
B951 1162BB             LD   DE,CDATA1
B954 CD88BA             CALL DISP
B957 05                 DEC  B
B958 D1                 POP  DE
B959 E1                 POP  HL
B95A C38AB9             JP   WAIT
                ;
B95D 78         CD2:    LD   A,B
B95E FE02               CP   02H
B960 C271B9             JP   NZ,CD3
B963 E5                 PUSH HL
B964 D5                 PUSH DE
B965 11CEBB             LD   DE,CDATA2
B968 CD88BA             CALL DISP
B96B 05                 DEC  B
B96C D1                 POP  DE
B96D E1                 POP  HL
B96E C38AB9             JP   WAIT
                ;
```

```
B971 E5       CD3:    PUSH HL
B972 D5               PUSH DE
B973 113ABC           LD   DE,CDATA3
B976 CD88BA           CALL DISP
B979 CD79BA           CALL WAT
B97C D1               POP  DE
B97D E1               POP  HL
B97E E5               PUSH HL
B97F D5               PUSH DE
B980 11CEBB           LD   DE,CDATA2
B983 CD88BA           CALL DISP
B986 0603             LD   B,03H
B988 D1               POP  DE
B989 E1               POP  HL
              ;
B98A CD79BA   WAIT:   CALL WAT
              ;
B98D C39EB9           JP   KIY
              ;
B990 DB09     WKIY:   IN   A,(09H)
B992 E601             AND  01H
B994 C249B9           JP   NZ,KAITEN
              ;
B997 ED7BA6BC         LD   SP,(SPSAVE)
              ;
B99B C326E6           JP   MON
              ;
B99E C5       KIY:    PUSH BC
B99F DB00             IN   A,(00H)          ;1
B9A1 47               LD   B,A
B9A2 E602             AND  02H
B9A4 CAD7B9           JP   Z,LDOWN
              ;
B9A7 78               LD   A,B              ;2
B9A8 E604             AND  04H
B9AA CAF0B9           JP   Z,DOWN
              ;
B9AD 78               LD   A,B              ;3
B9AE E608             AND  08H
B9B0 CA00BA           JP   Z,RDOWN
              ;
              ;
B9B3 78               LD   A,B              ;4
B9B4 E610             AND  10H
B9B6 CA19BA           JP   Z,LEFT
              ;
B9B9 78               LD   A,B              ;6
B9BA E640             AND  40H
B9BC CA25BA           JP   Z,RIGHT
              ;
B9BF 78               LD   A,B              ;7
B9C0 E680             AND  80H
B9C2 CA31BA           JP   Z,LUP
              ;

B9C5 DB01             IN   A,(01H)          ;8
B9C7 47               LD   B,A
B9C8 E601             AND  01H
B9CA CA4CBA           JP   Z,UP
              ;
B9CD 78               LD   A,B              ;9
B9CE E602             AND  02H
B9D0 CA5EBA           JP   Z,RUP
              ;
B9D3 C1       KIYPOP:POP   BC
              ;
B9D4 C390B9           JP   WKIY
              ;
```

```
B9D7 7B        LDOWN:  LD    A,E
B9D8 FEB5              CP    0B5H
B9DA CAD3B9            JP    Z,KIYPOP
B9DD 7A                LD    A,D
B9DE FE00              CP    00H
B9E0 CAD3B9            JP    Z,KIYPOP
B9E3 C5                PUSH  BC
B9E4 015000            LD    BC,0050H
B9E7 09                ADD   HL,BC
B9E8 C1                POP   BC
B9E9 1C                INC   E
B9EA CD02BB            CALL  LSM
B9ED C3D3B9            JP    KIYPOP
               ;
B9F0 7B        DOWN:   LD    A,E
B9F1 FEB5              CP    0B5H
B9F3 CAD3B9            JP    Z,KIYPOP
B9F6 C5                PUSH  BC
B9F7 015000            LD    BC,0050H
B9FA 09                ADD   HL,BC
B9FB C1                POP   BC
B9FC 1C                INC   E
B9FD C3D3B9            JP    KIYPOP
               ;
BA00 7B        RDOWN:  LD    A,E
BA01 FEB5              CP    0B5H
BA03 CAD3B9            JP    Z,KIYPOP
BA06 7A                LD    A,D
BA07 FE4A              CP    4AH
BA09 CAD3B9            JP    Z,KIYPOP
BA0C C5                PUSH  BC
BA0D 015000            LD    BC,0050H
BA10 09                ADD   HL,BC
BA11 C1                POP   BC
BA12 1C                INC   E
BA13 CDA7BA            CALL  RSM
BA16 C3D3B9            JP    KIYPOP
               ;
BA19 7A        LEFT:   LD    A,D
BA1A FE00              CP    00H
BA1C CAD3B9            JP    Z,KIYPOP
BA1F CD02BB            CALL  LSM
BA22 C3D3B9            JP    KIYPOP
               ;
BA25 7A        RIGHT:  LD    A,D
BA26 FE4A              CP    4AH
BA28 CAD3B9            JP    Z,KIYPOP
BA2B CDA7BA            CALL  RSM
BA2E C3D3B9            JP    KIYPOP
               ;
BA31 7B        LUP:    LD    A,E
BA32 FE00              CP    00H
BA34 CAD3B9            JP    Z,KIYPOP
BA37 7A                LD    A,D
BA38 FE00              CP    00H
BA3A CAD3B9            JP    Z,KIYPOP
BA3D C5                PUSH  BC
BA3E 015000            LD    BC,50H
BA41 A7                AND   A
BA42 ED42              SBC   HL,BC
BA44 C1                POP   BC
BA45 1D                DEC   E
BA46 CD02BB            CALL  LSM
BA49 C3D3B9            JP    KIYPOP
               ;
BA4C 7B        UP:     LD    A,E
BA4D FE00              CP    00H
BA4F CAD3B9            JP    Z,KIYPOP
BA52 C5                PUSH  BC
```

```
BA53 015000            LD    BC,0050H
BA56 A7                AND   A
BA57 ED42              SBC   HL,BC
BA59 C1                POP   BC
BA5A 1D                DEC   E
BA5B C3D3B9            JP    KIYPOP
                ;
BA5E 7B         RUP:   LD    A,E
BA5F FE00              CP    00H
BA61 CAD3B9            JP    Z,KIYPOP
BA64 7A                LD    A,D
BA65 FE4A              CP    4AH
BA67 CAD3B9            JP    Z,KIYPOP
BA6A C5                PUSH  BC
BA6B 015000            LD    BC,0050H
BA6E A7                AND   A
BA6F ED42              SBC   HL,BC
BA71 C1                POP   BC
BA72 1D                DEC   E
BA73 CDA7BA            CALL  RSM
BA76 C3D3B9            JP    KIYPOP
                ;
                ;
BA79 D5         WAT:   PUSH  DE                  ;ジカン マチ
BA7A 1602              LD    D,02H
BA7C 1EAF       W2:    LD    E,0AFH
BA7E 1D         W1:    DEC   E
BA7F C27EBA            JP    NZ,W1
BA82 15                DEC   D
BA83 C27CBA            JP    NZ,W2
BA86 D1                POP   DE
BA87 C9                RET
                ;
                ;
BA88 F3         DISP:  DI                        ;グラフ ノ ヒョウジ ルーチン ワリコミ NO
BA89 D35E              OUT   (GRAM2),A           ;カラー ノ イロ グリーン
BA8B C5                PUSH  BC
BA8C 0E12              LD    C,12H
BA8E 0606       LOP2:  LD    B,06H
BA90 1A         LOP1:  LD    A,(DE)
BA91 77                LD    (HL),A
BA92 23                INC   HL
BA93 13                INC   DE
BA94 05                DEC   B
BA95 C290BA            JP    NZ,LOP1
BA98 C5                PUSH  BC
BA99 014A00            LD    BC,004AH
BA9C 09                ADD   HL,BC
BA9D C1                POP   BC
BA9E 0D                DEC   C
BA9F C28EBA            JP    NZ,LOP2
BAA2 C1                POP   BC
BAA3 D35F              OUT   (MRAM),A            ;メイン メモリー ニ スル
BAA5 FB                EI                        ;ワリコミ OK
                ;
BAA6 C9                RET
                ;
BAA7 E5         RSM:   PUSH  HL                  ;RIGHT BIT
BAA8 2161BB            LD    HL,SSCD
BAAB 7E                LD    A,(HL)
BAAC FE10              CP    16
BAAE C2D4BA            JP    NZ,RSR1
BAB1 D5                PUSH  DE
BAB2 1608              LD    D,08H
BAB4 2162BB     RSM1:  LD    HL,CDATA1
BAB7 CD44BB            CALL  LRSM
BABA 21CEBB            LD    HL,CDATA2
BABD CD44BB            CALL  LRSM
BAC0 213ABC            LD    HL,CDATA3
```

```
BAC3 CD44BB          CALL LRSM
BAC6 15              DEC  D
BAC7 C2B4BA          JP   NZ,RSM1
BACA D1              POP  DE
BACB E1              POP  HL
BACC 14              INC  D
BACD 23              INC  HL
BACE 3E08            LD   A,08H
BAD0 3261BB          LD   (SSCD),A
BAD3 C9              RET
                ;
BAD4 34         RSR1: INC  (HL)
BAD5 2162BB          LD   HL,CDATA1
BAD8 CDE9BA          CALL RRSM
BADB 21CEBB          LD   HL,CDATA2
BADE CDE9BA          CALL RRSM
BAE1 213ABC          LD   HL,CDATA3
BAE4 CDE9BA          CALL RRSM
BAE7 E1              POP  HL
BAE8 C9              RET
                ;
BAE9 C5         RRSM: PUSH BC
BAEA 0E12            LD   C,18
BAEC 7E         RBA1: LD   A,(HL)
BAED CB3F            SRL  A
BAEF 77              LD   (HL),A
BAF0 23              INC  HL
BAF1 0605            LD   B,05H
BAF3 7E         RBB1: LD   A,(HL)
BAF4 CB1F            RR   A
BAF6 77              LD   (HL),A
BAF7 23              INC  HL
BAF8 05              DEC  B
BAF9 C2F3BA          JP   NZ,RBB1
BAFC 0D              DEC  C
BAFD C2ECBA          JP   NZ,RBA1
BB00 C1              POP  BC
BB01 C9              RET
                ;
BB02 E5         LSM:  PUSH HL                      ;LEFT BIT
BB03 2161BB          LD   HL,SSCD
BB06 7E              LD   A,(HL)
BB07 FE00            CP   00
BB09 C22FBB          JP   NZ,LSR1
BB0C D5              PUSH DE
BB0D 1608            LD   D,08H
BB0F 2162BB     LSM1: LD   HL,CDATA1
BB12 CDE9BA          CALL RRSM
BB15 21CEBB          LD   HL,CDATA2
BB18 CDE9BA          CALL RRSM
BB1B 213ABC          LD   HL,CDATA3
BB1E CDE9BA          CALL RRSM
BB21 15              DEC  D
BB22 C20FBB          JP   NZ,LSM1
BB25 D1              POP  DE
BB26 E1              POP  HL
BB27 2B              DEC  HL
BB28 15              DEC  D
BB29 3E08            LD   A,08H
BB2B 3261BB          LD   (SSCD),A
BB2E C9              RET
                ;
BB2F 35         LSR1: DEC  (HL)
BB30 2162BB          LD   HL,CDATA1
BB33 CD44BB          CALL LRSM
BB36 21CEBB          LD   HL,CDATA2
BB39 CD44BB          CALL LRSM
BB3C 213ABC          LD   HL,CDATA3
BB3F CD44BB          CALL LRSM
```

```
BB42 E1                 POP  HL
BB43 C9                 RET
                ;
BB44 C5         LRSM:   PUSH BC
BB45 016B00             LD   BC,107
BB48 09                 ADD  HL,BC
BB49 0E12               LD   C,18
BB4B 7E         LBA1:   LD   A,(HL)
BB4C CB27               SLA  A
BB4E 77                 LD   (HL),A
BB4F 2B                 DEC  HL
BB50 0605               LD   B,05H
BB52 7E         LBB1:   LD   A,(HL)
BB53 CB17               RL   A
BB55 77                 LD   (HL),A
BB56 2B                 DEC  HL
BB57 05                 DEC  B
BB58 C252BB             JP   NZ,LBB1
BB5B 0D                 DEC  C
BB5C C24BBB             JP   NZ,LBA1
BB5F C1                 POP  BC
BB60 C9                 RET
                ;
                ;
BB61            SSCD:   DS   1                    ;ｽｸﾛｰﾙ ﾄﾞｯﾄ ｽｳ
                ;
                ;
BB62 00000000   CDATA1:DB    00H,00H,00H,00H,00H,00H
BB66 0000
BB68 00001800           DB   00H,00H,18H,00H,00H,00H
BB6C 0000
BB6E 00FFFFFF           DB   00H,0FFH,0FFH,0FFH,00H,00H
BB72 0000
BB74 00FFFFFF           DB   00H,0FFH,0FFH,0FFH,07H,00H
BB78 0700
BB7A 00001800           DB   00H,00H,18H,00H,0FH,00H
BB7E 0F00
BB80 0001FFC0           DB   00H,01H,0FFH,0C0H,1FH,00H
BB84 1F00
BB86 00031FE0           DB   00H,03H,1FH,0E0H,3FH,00H
BB8A 3F00
BB8C 000C1FFF           DB   00H,0CH,1FH,0FFH,0FFH,00H
BB90 FF00
BB92 00301FFF           DB   00H,30H,1FH,0FFH,0FFH,00H
BB96 FF00
BB98 00C01FFF           DB   00H,0C0H,1FH,0FFH,0FEH,00H
BB9C FE00
BB9E 00FFFFE0           DB   00H,0FFH,0FFH,0E0H,00H,00H
BBA2 0000
BBA4 00FFFFC0           DB   00H,0FFH,0FFH,0C0H,00H,00H
BBA8 0000
BBAA 00FFFF80           DB   00H,0FFH,0FFH,80H,00H,00H
BBAE 0000
BBB0 003FFC00           DB   00H,3FH,0FCH,00H,00H,00H
BBB4 0000
BBB6 00030C00           DB   00H,03H,0CH,00H,00H,00H
BBBA 0000
BBBC 00C30C00           DB   00H,0C3H,0CH,00H,00H,00H
BBC0 0000
BBC2 007FFFE0           DB   00H,7FH,0FFH,0E0H,00H,00H
BBC6 0000
BBC8 00000000           DB   00H,00H,00H,00H,00H,00H
BBCC 0000
                ;
                ;
                ;
BBCE 00000000   CDATA2:DB    00H,00H,00H,00H,00H,00H
BBD2 0000
```

```
BBD4 00001800         DB    00H,00H,18H,00H,00H,00H
BBD8 0000
BBDA 0000FF00         DB    00H,00H,0FFH,00H,00H,00H
BBDE 0000
BBE0 0000FF00         DB    00H,00H,0FFH,00H,07H,00H
BBE4 0700
BBE6 00001800         DB    00H,00H,18H,00H,0FH,00H
BBEA 0F00
BBEC 0001FFC0         DB    00H,01H,0FFH,0C0H,1FH,00H
BBF0 1F00
BBF2 00031FE0         DB    00H,03H,1FH,0E0H,3FH,00H
BBF6 3F00
BBF8 000C1FFF         DB    00H,0CH,1FH,0FFH,0FFH,00H
BBFC FF00
BBFE 00301FFF         DB    00H,30H,1FH,0FFH,0FFH,00H
BC02 FF00
BC04 00C01FFF         DB    00H,0C0H,1FH,0FFH,0FEH,00H
BC08 FE00
BC0A 00FFFFE0         DB    00H,0FFH,0FFH,0E0H,00H,00H
BC0E 0000
BC10 00FFFFC0         DB    00H,0FFH,0FFH,0C0H,00H,00H
BC14 0000
BC16 00FFFF80         DB    00H,0FFH,0FFH,80H,00H,00H
BC1A 0000
BC1C 003FFC00         DB    00H,3FH,0FCH,00H,00H,00H
BC20 0000
BC22 00030C00         DB    00H,03H,0CH,00H,00H,00H
BC26 0000
BC28 00C30C00         DB    00H,0C3H,0CH,00H,00H,00H
BC2C 0000
BC2E 007FFFE0         DB    00H,7FH,0FFH,0E0H,00H,00H
BC32 0000
BC34 00000000         DB    00H,00H,00H,00H,00H,00H
BC38 0000
                  ;
BC3A 00000000 CDATA3:DB    00H,00H,00H,00H,00H,00H
BC3E 0000
BC40 00001800         DB    00H,00H,18H,00H,00H,00H
BC44 0000
BC46 00001800         DB    00H,00H,18H,00H,00H,00H
BC4A 0000
BC4C 00001800         DB    00H,00H,18H,00H,07H,00H
BC50 0700
BC52 00001800         DB    00H,00H,18H,00H,0FH,00H
BC56 0F00
BC58 0001FFC0         DB    00H,01H,0FFH,0C0H,1FH,00H
BC5C 1F00
BC5E 00031FE0         DB    00H,03H,1FH,0E0H,3FH,00H
BC62 3F00
BC64 000C1FFF         DB    00H,0CH,1FH,0FFH,0FFH,00H
BC68 FF00
BC6A 00301FFF         DB    00H,30H,1FH,0FFH,0FFH,00H
BC6E FF00
BC70 00C01FFF         DB    00H,0C0H,1FH,0FFH,0FEH,00H
BC74 FE00
BC76 00FFFFE0         DB    00H,0FFH,0FFH,0E0H,00H,00H
BC7A 0000
BC7C 00FFFFC0         DB    00H,0FFH,0FFH,0C0H,00H,00H
BC80 0000
BC82 00FFFF80         DB    00H,0FFH,0FFH,80H,00H,00H
BC86 0000
BC88 003FFC00         DB    00H,3FH,0FCH,00H,00H,00H
BC8C 0000
BC8E 00030C00         DB    00H,03H,0CH,00H,00H,00H
BC92 0000
BC94 00C30C00         DB    00H,0C3H,0CH,00H,00H,00H
BC98 0000
BC9A 007FFFE0         DB    00H,7FH,0FFH,0E0H,00H,00H
```

```
BC9E 0000
BCA0 00000000         DB    00H,00H,00H,00H,00H,00H
BCA4 0000
                ;
                ;
BCA6            SPSAVE:DS    02H
BCA8                   DS    60
BCE4            STACK:
                ;
BCE4                   END
```

## テキスト画面とグラフィックス画面を重ね合わせる

PC-8801mkIISRは，テキスト画面とグラフィックス画面の重ね合わせが簡単です．**図5-15**のような画面を作ってみましょう．

テキスト文字は8ビット×8ビットで構成されているので，縦の8ビット分がグラフィックス画面と重なります．グラフィックス画面がテキストの文字までいかないように，グラフィックス画面の縦のドット数を数えているEレジスタから8ビット引いておきます．テキスト文字のカラー化は第2章（P.93）でやっていますので思い出してください．テキスト画面とグラフィックス画面の重ね合わせのプログラムを**リスト5-2**に示します．**リスト5-1**にテキスト文字を表示する部分を追加しただけです．テキストモードを同時に使用することができるので，ゲームなどのスコアやメッセージを表示する場合に便利です．

**図5-15　テキスト画面との重ね合わせ**

**●リスト5-2 テキスト画面との重ね合わせ**

```
                  ;**************************************************
                  ;
                  ;GRPH--9 ﾖｺ ﾉ ｽﾑｰｽﾞ ｲﾄﾞｳ  ﾄ ﾃｷｽﾄ ｶﾞﾒﾝ ﾉ ｶｻﾈｱﾜｾ
                  ;
                  ;PROGRAM NAME ---> Grph9.e
                  ;
                  ;**************************************************
                  ;
B900              START:  EQU   0B900H
C000              GRAMS:  EQU   0C000H
6F6B              COLOR:  EQU   6F6BH
428B              CLS:    EQU   428BH              ;ｶｰｿﾙ ｹｽ
```

```
E626            MON:    EQU   0E626H
005C            GRAM0:  EQU   5CH
005D            GRAM1:  EQU   5DH
005E            GRAM2:  EQU   5EH
005F            MRAM:   EQU   5FH
                ;
                        ORG   START
                ;
B900 ED73DEBC           LD    (SPSAVE),SP        ;スタック ノ タイヒ
B904 311CBD             LD    SP,STACK
                ;
B907 011950             LD    BC,5019H
B90A 3E01               LD    A,01H
B90C 32B2E6             LD    (0E6B2H),A
B90F 3E19               LD    A,19H
B911 32B3E6             LD    (0E6B3H),A
B914 3EE8               LD    A,0E8H
B916 32B4E6             LD    (0E6B4H),A
B919 3E00               LD    A,00H
B91B 32B8E6             LD    (0E6B8H),A
B91E 3EFF               LD    A,0FFH
B920 32B9E6             LD    (0E6B9H),A
B923 CD6B6F             CALL  COLOR
                ;
B926 CD8B42             CALL  CLS
                ;
B929 DD2118F4           LD    IX,0F418H          ;テキスト モシ゛ ノ ヒョウシ゛
B92D DD360000           LD    (IX+0),00H
B931 DD3601C8           LD    (IX+1),0C8H        ; テキスト ノ カラー キイロ
                ;
B935 21C1BC             LD    HL,TEDATA
B938 11E0F3             LD    DE,0F3E0H          ;テキスト ノ ヒョウシ゛ イチ
B93B 011C00             LD    BC,28
B93E EDB0               LDIR
                ;
B940 3E08       MAIN:   LD    A,08H
B942 327CBB             LD    (SSCD),A
                ;
B945 2100C0             LD    HL,GRAMS
B948 110000             LD    DE,0000H           ;ハシ゛メノ ヒョウシ゛イチ
B94B 3E27               LD    A,27H
B94D 23         ML1:    INC   HL
B94E 14                 INC   D
B94F 3D                 DEC   A
B950 C24DB9             JP    NZ,ML1
B953 3E64               LD    A,100
B955 015000             LD    BC,0050H
B958 09         ML2:    ADD   HL,BC
B959 1C                 INC   E
B95A 3D                 DEC   A
B95B C258B9             JP    NZ,ML2
B95E 7B                 LD    A,E                ; TEXT ノ ト゛ット フ゛ン ヒク
B95F D608               SUB   08H
B961 5F                 LD    E,A
                ;
B962 0603               LD    B,03H
                ;
B964 78         KAITEN:LD     A,B
B965 FE03               CP    03H
B967 C278B9             JP    NZ,CD2
```

```
B96A E5                PUSH HL
B96B D5                PUSH DE
B96C 117DBB            LD   DE,CDATA1
B96F CDA3BA            CALL DISP
B972 05                DEC  B
B973 D1                POP  DE
B974 E1                POP  HL
B975 C3A5B9            JP   WAIT
                  ;
B978 78           CD2: LD   A,B
B979 FE02              CP   02H
B97B C28CB9            JP   NZ,CD3
B97E E5                PUSH HL
B97F D5                PUSH DE
B980 11E9BB            LD   DE,CDATA2
B983 CDA3BA            CALL DISP
B986 05                DEC  B
B987 D1                POP  DE
B988 E1                POP  HL
B989 C3A5B9            JP   WAIT
                  ;
B98C E5           CD3: PUSH HL
B98D D5                PUSH DE
B98E 1155BC            LD   DE,CDATA3
B991 CDA3BA            CALL DISP
B994 CD94BA            CALL WAT
B997 D1                POP  DE
B998 E1                POP  HL
B999 E5                PUSH HL
B99A D5                PUSH DE
B99B 11E9BB            LD   DE,CDATA2
B99E CDA3BA            CALL DISP
B9A1 0603              LD   B,03H
B9A3 D1                POP  DE
B9A4 E1                POP  HL
                  ;
B9A5 CD94BA       WAIT: CALL WAT
                  ;
B9A8 C3B9B9            JP   KIY
                  ;
B9AB DB09         WKIY: IN  A,(09H)
B9AD E601              AND  01H
B9AF C264B9            JP   NZ,KAITEN
                  ;
B9B2 ED7BDEBC          LD   SP,(SPSAVE)
                  ;
B9B6 C326E6            JP   MON
                  ;
B9B9 C5           KIY: PUSH BC
B9BA DB00              IN   A,(00H)          ;1
B9BC 47                LD   B,A
B9BD E602              AND  02H
B9BF CAF2B9            JP   Z,LDOWN
                  ;
B9C2 78                LD   A,B              ;2
B9C3 E604              AND  04H
B9C5 CA0BBA            JP   Z,DOWN
                  ;
B9C8 78                LD   A,B              ;3
B9C9 E608              AND  08H
```

```
B9CB CA1BBA              JP    Z,RDOWN
                 ;
                 ;
B9CE 78                  LD    A,B            ;4
B9CF E610                AND   10H
B9D1 CA34BA              JP    Z,LEFT
                 ;
B9D4 78                  LD    A,B            ;6
B9D5 E640                AND   40H
B9D7 CA40BA              JP    Z,RIGHT
                 ;
B9DA 78                  LD    A,B            ;7
B9DB E680                AND   80H
B9DD CA4CBA              JP    Z,LUP
                 ;


B9E0 DB01                IN    A,(01H)        ;8
B9E2 47                  LD    B,A
B9E3 E601                AND   01H
B9E5 CA67BA              JP    Z,UP
                 ;
B9E8 78                  LD    A,B            ;9
B9E9 E602                AND   02H
B9EB CA79BA              JP    Z,RUP
                 ;
B9EE C1          KIYPOP:POP    BC
                 ;
B9EF C3ABB9              JP    WKIY
                 ;
B9F2 7B          LDOWN:  LD    A,E
B9F3 FEAE                CP    0AEH
B9F5 CAEEB9              JP    Z,KIYPOP
B9F8 7A                  LD    A,D
B9F9 FE00                CP    00H
B9FB CAEEB9              JP    Z,KIYPOP
B9FE C5                  PUSH  BC
B9FF 015000              LD    BC,0050H
BA02 09                  ADD   HL,BC
BA03 C1                  POP   BC
BA04 1C                  INC   E
BA05 CD1DBB              CALL  LSM
BA08 C3EEB9              JP    KIYPOP
                 ;
BA0B 7B          DOWN:   LD    A,E
BA0C FEAE                CP    0AEH
BA0E CAEEB9              JP    Z,KIYPOP
BA11 C5                  PUSH  BC
BA12 015000              LD    BC,0050H
BA15 09                  ADD   HL,BC
BA16 C1                  POP   BC
BA17 1C                  INC   E
BA18 C3EEB9              JP    KIYPOP
                 ;
BA1B 7B          RDOWN:  LD    A,E
BA1C FEAE                CP    0AEH
BA1E CAEEB9              JP    Z,KIYPOP
BA21 7A                  LD    A,D
BA22 FE4A                CP    4AH
BA24 CAEEB9              JP    Z,KIYPOP
BA27 C5                  PUSH  BC
```

```
BA28 015000          LD   BC,0050H
BA2B 09              ADD  HL,BC
BA2C C1              POP  BC
BA2D 1C              INC  E
BA2E CDC2BA          CALL RSM
BA31 C3EEB9          JP   KIYPOP
                ;
BA34 7A         LEFT: LD  A,D
BA35 FE00            CP   00H
BA37 CAEEB9          JP   Z,KIYPOP
BA3A CD1DBB          CALL LSM
BA3D C3EEB9          JP   KIYPOP
                ;
BA40 7A         RIGHT: LD A,D
BA41 FE4A            CP   4AH
BA43 CAEEB9          JP   Z,KIYPOP
BA46 CDC2BA          CALL RSM
BA49 C3EEB9          JP   KIYPOP
                ;
BA4C 7B         LUP:  LD  A,E
BA4D FE00            CP   00H
BA4F CAEEB9          JP   Z,KIYPOP
BA52 7A              LD   A,D
BA53 FE00            CP   00H
BA55 CAEEB9          JP   Z,KIYPOP
BA58 C5              PUSH BC
BA59 015000          LD   BC,50H
BA5C A7              AND  A
BA5D ED42            SBC  HL,BC
BA5F C1              POP  BC
BA60 1D              DEC  E
BA61 CD1DBB          CALL LSM
BA64 C3EEB9          JP   KIYPOP
                ;
BA67 7B         UP:   LD  A,E
BA68 FE00            CP   00H
BA6A CAEEB9          JP   Z,KIYPOP
BA6D C5              PUSH BC
BA6E 015000          LD   BC,0050H
BA71 A7              AND  A
BA72 ED42            SBC  HL,BC
BA74 C1              POP  BC
BA75 1D              DEC  E
BA76 C3EEB9          JP   KIYPOP
                ;
BA79 7B         RUP:  LD  A,E
BA7A FE00            CP   00H
BA7C CAEEB9          JP   Z,KIYPOP
BA7F 7A              LD   A,D
BA80 FE4A            CP   4AH
BA82 CAEEB9          JP   Z,KIYPOP
BA85 C5              PUSH BC
BA86 015000          LD   BC,0050H
BA89 A7              AND  A
BA8A ED42            SBC  HL,BC
BA8C C1              POP  BC
BA8D 1D              DEC  E
BA8E CDC2BA          CALL RSM
BA91 C3EEB9          JP   KIYPOP
                ;
```

```
                   ;
BA94 D5         WAT:    PUSH DE             ;ｼﾞｶﾝ ﾏﾁ
BA95 1602               LD   D,02H
BA97 1EAF       W2:     LD   E,0AFH
BA99 1D         W1:     DEC  E
BA9A C299BA             JP   NZ,W1
BA9D 15                 DEC  D
BA9E C297BA             JP   NZ,W2
BAA1 D1                 POP  DE
BAA2 C9                 RET
                ;
                ;
BAA3 F3         DISP:   DI                  ;ｸﾞﾗﾌ ﾉ ﾋｮｳｼﾞ ﾙｰﾁﾝ ﾜﾘｺﾐ NO
BAA4 D35E               OUT  (GRAM2),A      ;ｶﾗｰ ﾉ ｲﾛ ｸﾞﾘｰﾝ
BAA6 C5                 PUSH BC
BAA7 0E12               LD   C,12H
BAA9 0606       LOP2:   LD   B,06H
BAAB 1A         LOP1:   LD   A,(DE)
BAAC 77                 LD   (HL),A
BAAD 23                 INC  HL
BAAE 13                 INC  DE
BAAF 05                 DEC  B
BAB0 C2ABBA             JP   NZ,LOP1
BAB3 C5                 PUSH BC
BAB4 014A00             LD   BC,004AH
BAB7 09                 ADD  HL,BC
BAB8 C1                 POP  BC
BAB9 0D                 DEC  C
BABA C2A9BA             JP   NZ,LOP2
BABD C1                 POP  BC
BABE D35F               OUT  (MRAM),A       ;ﾒｲﾝ ﾒﾓﾘｰ ﾆ ｽﾙ
BAC0 FB                 EI                  ;ﾜﾘｺﾐ OK
                ;
BAC1 C9                 RET
                ;
BAC2 E5         RSM:    PUSH HL             ;RIGHT BIT
BAC3 217CBB             LD   HL,SSCD
BAC6 7E                 LD   A,(HL)
BAC7 FE10               CP   16
BAC9 C2EFBA             JP   NZ,RSR1
BACC D5                 PUSH DE
BACD 1608               LD   D,08H
BACF 217DBB     RSM1:   LD   HL,CDATA1
BAD2 CD5FBB             CALL LRSM
BAD5 21E9BB             LD   HL,CDATA2
BAD8 CD5FBB             CALL LRSM
BADB 2155BC             LD   HL,CDATA3
BADE CD5FBB             CALL LRSM
BAE1 15                 DEC  D
BAE2 C2CFBA             JP   NZ,RSM1
BAE5 D1                 POP  DE
BAE6 E1                 POP  HL
BAE7 14                 INC  D
BAE8 23                 INC  HL
BAE9 3E08               LD   A,08H
BAEB 327CBB             LD   (SSCD),A
BAEE C9                 RET
                ;
BAEF 34         RSR1:   INC  (HL)
BAF0 217DBB             LD   HL,CDATA1
BAF3 CD04BB             CALL RRSM
```

```
BAF6 21E9BB            LD    HL,CDATA2
BAF9 CD04BB            CALL  RRSM
BAFC 2155BC            LD    HL,CDATA3
BAFF CD04BB            CALL  RRSM
BB02 E1                POP   HL
BB03 C9                RET
                ;
BB04 C5         RRSM:  PUSH  BC
BB05 0E12              LD    C,18
BB07 7E         RBA1:  LD    A,(HL)
BB08 CB3F              SRL   A
BB0A 77                LD    (HL),A
BB0B 23                INC   HL
BB0C 0605              LD    B,05H
BB0E 7E         RBB1:  LD    A,(HL)
BB0F CB1F              RR    A
BB11 77                LD    (HL),A
BB12 23                INC   HL
BB13 05                DEC   B
BB14 C20EBB            JP    NZ,RBB1
BB17 0D                DEC   C
BB18 C207BB            JP    NZ,RBA1
BB1B C1                POP   BC
BB1C C9                RET
                ;
BB1D E5         LSM:   PUSH  HL                  ;LEFT BIT
BB1E 217CBB            LD    HL,SSCD
BB21 7E                LD    A,(HL)
BB22 FE00              CP    00
BB24 C24ABB            JP    NZ,LSR1
BB27 D5                PUSH  DE
BB28 1608              LD    D,08H
BB2A 217DBB     LSM1:  LD    HL,CDATA1
BB2D CD04BB            CALL  RRSM
BB30 21E9BB            LD    HL,CDATA2
BB33 CD04BB            CALL  RRSM
BB36 2155BC            LD    HL,CDATA3
BB39 CD04BB            CALL  RRSM
BB3C 15                DEC   D
BB3D C22ABB            JP    NZ,LSM1
BB40 D1                POP   DE
BB41 E1                POP   HL
BB42 2B                DEC   HL
BB43 15                DEC   D
BB44 3E08              LD    A,08H
BB46 327CBB            LD    (SSCD),A
BB49 C9                RET
                ;
BB4A 35         LSR1:  DEC   (HL)
BB4B 217DBB            LD    HL,CDATA1
BB4E CD5FBB            CALL  LRSM
BB51 21E9BB            LD    HL,CDATA2
BB54 CD5FBB            CALL  LRSM
BB57 2155BC            LD    HL,CDATA3
BB5A CD5FBB            CALL  LRSM
BB5D E1                POP   HL
BB5E C9                RET
                ;
BB5F C5         LRSM:  PUSH  BC
BB60 016B00            LD    BC,107
BB63 09                ADD   HL,BC
```

```
BB64 0E12               LD   C,18
BB66 7E        LBA1:    LD   A,(HL)
BB67 CB27               SLA  A
BB69 77                 LD   (HL),A
BB6A 2B                 DEC  HL
BB6B 0605               LD   B,05H
BB6D 7E        LBB1:    LD   A,(HL)
BB6E CB17               RL   A
BB70 77                 LD   (HL),A
BB71 2B                 DEC  HL
BB72 05                 DEC  B
BB73 C26DBB             JP   NZ,LBB1
BB76 0D                 DEC  C
BB77 C266BB             JP   NZ,LBA1
BB7A C1                 POP  BC
BB7B C9                 RET
               ;
               ;
BB7C           SSCD:    DS   1                          ;ｽｸﾛｰﾙ ﾄﾞｯﾄ ｽｳ
               ;
               ;
BB7D 00000000  CDATA1:DB     00H,00H,00H,00H,00H,00H
BB81 0000
BB83 00001800           DB   00H,00H,18H,00H,00H,00H
BB87 0000
BB89 00FFFFFF           DB   00H,0FFH,0FFH,0FFH,00H,00H
BB8D 0000
BB8F 00FFFFFF           DB   00H,0FFH,0FFH,0FFH,07H,00H
BB93 0700
BB95 00001800           DB   00H,00H,18H,00H,0FH,00H
BB99 0F00
BB9B 0001FFC0           DB   00H,01H,0FFH,0C0H,1FH,00H
BB9F 1F00
BBA1 00031FE0           DB   00H,03H,1FH,0E0H,3FH,00H
BBA5 3F00
BBA7 000C1FFF           DB   00H,0CH,1FH,0FFH,0FFH,00H
BBAB FF00
BBAD 00301FFF           DB   00H,30H,1FH,0FFH,0FFH,00H
BBB1 FF00
BBB3 00C01FFF           DB   00H,0C0H,1FH,0FFH,0FEH,00H
BBB7 FE00
BBB9 00FFFFE0           DB   00H,0FFH,0FFH,0E0H,00H,00H
BBBD 0000
BBBF 00FFFFC0           DB   00H,0FFH,0FFH,0C0H,00H,00H
BBC3 0000
BBC5 00FFFF80           DB   00H,0FFH,0FFH,80H,00H,00H
BBC9 0000
BBCB 003FFC00           DB   00H,3FH,0FCH,00H,00H,00H
BBCF 0000
BBD1 00030C00           DB   00H,03H,0CH,00H,00H,00H
BBD5 0000
BBD7 00C30C00           DB   00H,0C3H,0CH,00H,00H,00H
BBDB 0000
BBDD 007FFFE0           DB   00H,7FH,0FFH,0E0H,00H,00H
BBE1 0000
BBE3 00000000           DB   00H,00H,00H,00H,00H,00H
BBE7 0000
               ;
               ;
               ;
BBE9 00000000  CDATA2:DB     00H,00H,00H,00H,00H,00H
```

```
BBED 0000
BBEF 00001800          DB   00H,00H,18H,00H,00H,00H
BBF3 0000
BBF5 0000FF00          DB   00H,00H,0FFH,00H,00H,00H
BBF9 0000
BBFB 0000FF00          DB   00H,00H,0FFH,00H,07H,00H
BBFF 0700
BC01 00001800          DB   00H,00H,18H,00H,0FH,00H
BC05 0F00
BC07 0001FFC0          DB   00H,01H,0FFH,0C0H,1FH,00H
BC0B 1F00
BC0D 00031FE0          DB   00H,03H,1FH,0E0H,3FH,00H
BC11 3F00
BC13 000C1FFF          DB   00H,0CH,1FH,0FFH,0FFH,00H
BC17 FF00
BC19 00301FFF          DB   00H,30H,1FH,0FFH,0FFH,00H
BC1D FF00
BC1F 00C01FFF          DB   00H,0C0H,1FH,0FFH,0FEH,00H
BC23 FE00
BC25 00FFFFE0          DB   00H,0FFH,0FFH,0E0H,00H,00H
BC29 0000
BC2B 00FFFFC0          DB   00H,0FFH,0FFH,0C0H,00H,00H
BC2F 0000
BC31 00FFFF80          DB   00H,0FFH,0FFH,80H,00H,00H
BC35 0000
BC37 003FFC00          DB   00H,3FH,0FCH,00H,00H,00H
BC3B 0000
BC3D 00030C00          DB   00H,03H,0CH,00H,00H,00H
BC41 0000
BC43 00C30C00          DB   00H,0C3H,0CH,00H,00H,00H
BC47 0000
BC49 007FFFE0          DB   00H,7FH,0FFH,0E0H,00H,00H
BC4D 0000
BC4F 00000000          DB   00H,00H,00H,00H,00H,00H
BC53 0000
                  ;
BC55 00000000 CDATA3:DB   00H,00H,00H,00H,00H,00H
BC59 0000
BC5B 00001800          DB   00H,00H,18H,00H,00H,00H
BC5F 0000
BC61 00001800          DB   00H,00H,18H,00H,00H,00H
BC65 0000
BC67 00001800          DB   00H,00H,18H,00H,07H,00H
BC6B 0700
BC6D 00001800          DB   00H,00H,18H,00H,0FH,00H
BC71 0F00
BC73 0001FFC0          DB   00H,01H,0FFH,0C0H,1FH,00H
BC77 1F00
BC79 00031FE0          DB   00H,03H,1FH,0E0H,3FH,00H
BC7D 3F00
BC7F 000C1FFF          DB   00H,0CH,1FH,0FFH,0FFH,00H
BC83 FF00
BC85 00301FFF          DB   00H,30H,1FH,0FFH,0FFH,00H
BC89 FF00
BC8B 00C01FFF          DB   00H,0C0H,1FH,0FFH,0FEH,00H
BC8F FE00
BC91 00FFFFE0          DB   00H,0FFH,0FFH,0E0H,00H,00H
BC95 0000
BC97 00FFFFC0          DB   00H,0FFH,0FFH,0C0H,00H,00H
BC9B 0000
```

```
BC9D 00FFFF80         DB    00H,0FFH,0FFH,80H,00H,00H
BCA1 0000
BCA3 003FFC00         DB    00H,3FH,0FCH,00H,00H,00H
BCA7 0000
BCA9 00030C00         DB    00H,03H,0CH,00H,00H,00H
BCAD 0000
BCAF 00C30C00         DB    00H,0C3H,0CH,00H,00H,00H
BCB3 0000
BCB5 007FFFE0         DB    00H,7FH,0FFH,0E0H,00H,00H
BCB9 0000
BCBB 00000000         DB    00H,00H,00H,00H,00H,00H
BCBF 0000
                ;
BCC1 47525048 TEDATA:DB    47H,52H,50H,48H,20H
BCC5 20
BCC6 54455854         DB    54H,45H,58H,54H,20H,4DH,49H,58H,20H
BCCA 204D4958
BCCE 20
BCCF 44454D4F         DB    44H,45H,4DH,4FH,20H,50H,52H,4FH
BCD3 2050524F
BCD7 4752414D         DB    47H,52H,41H,4DH,00H,00H,00H
BCDB 000000
                ;
                ;
BCDE          SPSAVE:DS    02H
BCE0                 DS    60
BD1C          STACK:
                ;
BD1C                 END
```

## DMAを止めて処理スピードを上げる

**リスト5-1，5-2**ではプログラムのアルゴリズムの関係でマシン語でも動きが非常に遅くなっていました．そこで，プログラムのアルゴリズムはそのままでスピードを上げてみましょう．PC-8801シリーズで処理スピードが遅くなっている原因はCRTC(μPD3301)へのデータ転送です．これはテキストモードのVRAMを読み込むのにDMA (Direct Memory Access) 処理をしているために，CPUがストップさせられてしまうためです．もし，DMA処理をしていなければ30%以上のスピードアップをはかることができます．ただし，テキスト画面を使用する場合にはこのままでは使用できません．

```
DMAのストップ────→    LD   A, 00H
                      OUT  (51H), A

DMAの再スタート───→   LD   BC, 5019H
                      LD   A, 01H
                      LD   (0E6B2H), A
```

```
LD  A, 19H
LD  (0E6B3H), A
LD  A, 0E8H
LD  (0E6B4H), A
LD  A, 00H
LD  (0E6B8H), A
LD  A, 0FFH
LD  (0E6B9H), A
CALL  6F6BH
```

この2つのプログラムを初めと終りの方に入れるだけですから楽です．実際ゲームなどを作る場合は表示する文字からすべてグラフィックスを使用した方が出来上がったときの評価は高いですから，テキスト文字を使用しない方が賢明です．そのときにこのプログラムが役に立つでしょう．

この章で作った**リスト5-1**に追加をして処理スピードを上げてみました．今度は速すぎて回転翼のタイミングがとれません．WAIT ルーチンのデータを変えてスピードを遅くしてください．

では，プログラムを**リスト5-3**に示します．

これで第5章を終わります．グラフィックスの基本的なことは説明しました．この他に細かいことはたくさんありますが，今までの部分で説明できなかったことは次の「マシン語アラカルト」で説明していきたいと思います．

**●リスト5-3 スムーズな移動(DMA ストップ)**

```
        ;*************************************************************
        ;
        ;GRPH--8 ヨコ ノ スムース゛ イト゛ウ （ヘリコフ゜ター ノ ハネ ノ カイテン ト キーニュウリョク  8ホウコウ
        ;
        ;PROGRAM NAME ---> GM8.e  (DMA STOP)
        ;
        ;*************************************************************

B900            START:  EQU  0B900H
C000            GRAMS:  EQU  0C000H
6F6B            COLOR:  EQU  6F6BH
428B            CLS:    EQU  428BH                 ;カーソル ケス
E626            MON:    EQU  0E626H
005C            GRAM0:  EQU  5CH
005D            GRAM1:  EQU  5DH
005E            GRAM2:  EQU  5EH
005F            MRAM:   EQU  5FH
                ;
                        ORG  START
                ;
B900 ED73C9BC           LD   (SPSAVE).SP           ;スタック ノ タイヒ
B904 3107BD             LD   SP.STACK
```

```
                     ;
B907 011950             LD    BC.5019H
B90A 3E01               LD    A.01H
B90C 32B2E6             LD    (0E6B2H).A
B90F 3E19               LD    A.19H
B911 32B3E6             LD    (0E6B3H).A
B914 3EE8               LD    A.0E8H
B916 32B4E6             LD    (0E6B4H).A
B919 3E00               LD    A.00H
B91B 32B8E6             LD    (0E6B8H).A
B91E 3EFF               LD    A.0FFH
B920 32B9E6             LD    (0E6B9H).A
B923 CD6B6F             CALL  COLOR
                     ;
B926 CD8B42             CALL  CLS
                     ;
B929 3E00               LD    A.00H           ;DMA STOP } 追加した部分
B92B D351               OUT   (51H).A
                     ;
B92D 3E08            MAIN:  LD    A.08H
B92F 3284BB             LD    (SSCD).A
                     ;
B932 2100C0             LD    HL.GRAMS
B935 110000             LD    DE.0000H        ;ﾊｼﾞﾒﾉ ﾋｮｳｼﾞｲﾁ
B938 3E27               LD    A.27H
B93A 23              ML1:   INC   HL
B93B 14                 INC   D
B93C 3D                 DEC   A
B93D C23AB9             JP    NZ.ML1
B940 3E64               LD    A.100
B942 015000             LD    BC.0050H
B945 09              ML2:   ADD   HL.BC
B946 1C                 INC   E
B947 3D                 DEC   A
B948 C245B9             JP    NZ.ML2
                     ;
B94B 0603               LD    B.03H
                     ;
B94D 78              KAITEN:LD    A.B
B94E FE03               CP    03H
B950 C261B9             JP    NZ.CD2
B953 E5                 PUSH  HL
B954 D5                 PUSH  DE
B955 1185BB             LD    DE.CDATA1
B958 CDABBA             CALL  DISP
B95B 05                 DEC   B
B95C D1                 POP   DE
B95D E1                 POP   HL
B95E C38EB9             JP    WAIT
                     ;
B961 78              CD2:   LD    A.B
B962 FE02               CP    02H
B964 C275B9             JP    NZ.CD3
B967 E5                 PUSH  HL
B968 D5                 PUSH  DE
B969 11F1BB             LD    DE.CDATA2
B96C CDABBA             CALL  DISP
B96F 05                 DEC   B
B970 D1                 POP   DE
B971 E1                 POP   HL
B972 C38EB9             JP    WAIT
```

```
                    ;
B975 E5         CD3:    PUSH HL
B976 D5                 PUSH DE
B977 115DBC             LD   DE.CDATA3
B97A CDABBA             CALL DISP
B97D CD9CBA             CALL WAT
B980 D1                 POP  DE
B981 E1                 POP  HL
B982 E5                 PUSH HL
B983 D5                 PUSH DE
B984 11F1BB             LD   DE.CDATA2
B987 CDABBA             CALL DISP
B98A 0603               LD   B.03H
B98C D1                 POP  DE
B98D E1                 POP  HL
                ;
B98E CD9CBA     WAIT:   CALL WAT
                ;
B991 C3C1B9             JP   KIY
                ;
B994 DB09       WKIY:   IN   A.(09H)
B996 E601               AND  01H
B998 C24DB9             JP   NZ.KAITEN
                ;
B99B 011950             LD   BC.5019H          ;DMA START  ┐
B99E 3E01               LD   A.01H                         │
B9A0 32B2E6             LD   (0E6B2H).A                    │
B9A3 3E19               LD   A.19H                         │
B9A5 32B3E6             LD   (0E6B3H).A                    │
B9A8 3EE8               LD   A.0E8H                        │ 追加した
B9AA 32B4E6             LD   (0E6B4H).A                    │ 部分
B9AD 3E00               LD   A.00H                         │
B9AF 32B8E6             LD   (0E6B8H).A                    │
B9B2 3EFF               LD   A.0FFH                        │
B9B4 32B9E6             LD   (0E6B9H).A                    ┘
B9B7 CD6B6F             CALL COLOR
                ;
B9BA ED7BC9BC           LD   SP.(SPSAVE)
                ;
B9BE C326E6                  MON
                ;
B9C1 C5         KIY:    PUSH BC
B9C2 DB00               IN   A.(00H)
B9C4 47                 LD   B.A
B9C5 E602               AND  02H
B9C7 CAFAB9             JP   Z.LDOWN
                ;
B9CA 78                 LD   A.B               ;2
B9CB E604               AND  04H
B9CD CA13BA             JP   Z.DOWN
                ;
B9D0 78                 LD   A.B               ;3
B9D1 E608               AND  08H
B9D3 CA23BA             JP   Z.RDOWN
                ;
                ;
B9D6 78                 LD   A.B               ;4
B9D7 E610               AND  10H
B9D9 CA3CBA             JP   Z.LEFT
                ;
B9DC 78                 LD   A.B               ;6
```

```
B9DD E640          AND  40H
B9DF CA48BA        JP   Z.RIGHT
               ;
B9E2 78            LD   A.B             ;7
B9E3 E680          AND  80H
B9E5 CA54BA        JP   Z.LUP
               ;


B9E8 DB01          IN   A.(01H)         ;8
B9EA 47            LD   B.A
B9EB E601          AND  01H
B9ED CA6FBA        JP   Z.UP
               ;
B9F0 78            LD   A.B             ;9
B9F1 E602          AND  02H
B9F3 CA81BA        JP   Z.RUP
               ;
B9F6 C1      KIYPOP:POP  BC
               ;
B9F7 C394B9        JP   WKIY
               ;
B9FA 7B      LDOWN: LD   A.E
B9FB FEB5          CP   0B5H
B9FD CAF6B9        JP   Z.KIYPOP
BA00 7A            LD   A.D
BA01 FE00          CP   00H
BA03 CAF6B9        JP   Z.KIYPOP
BA06 C5            PUSH BC
BA07 015000        LD   BC.0050H
BA0A 09            ADD  HL.BC
BA0B C1            POP  BC
BA0C 1C            INC  E
BA0D CD25BB        CALL LSM
BA10 C3F6B9        JP   KIYPOP
               ;
BA13 7B      DOWN:  LD   A.E
BA14 FEB5          CP   0B5H
BA16 CAF6B9        JP   Z.KIYPOP
BA19 C5            PUSH BC
BA1A 015000        LD   BC.0050H
BA1D 09            ADD  HL.BC
BA1E C1            POP  BC
BA1F 1C            INC  E
BA20 C3F6B9        JP   KIYPOP
               ;
BA23 7B      RDOWN: LD   A.E
BA24 FEB5          CP   0B5H
BA26 CAF6B9        JP   Z.KIYPOP
BA29 7A            LD   A.D
BA2A FE4A          CP   4AH
BA2C CAF6B9        JP   Z.KIYPOP
BA2F C5            PUSH BC
BA30 015000        LD   BC.0050H
BA33 09            ADD  HL.BC
BA34 C1            POP  BC
BA35 1C            INC  E
BA36 CDCABA        CALL RSM
BA39 C3F6B9        JP   KIYPOP
               ;
BA3C 7A      LEFT:  LD   A.D
```

```
BA3D FE00             CP    00H
BA3F CAF6B9           JP    Z.KIYPOP
BA42 CD25BB           CALL  LSM
BA45 C3F6B9           JP    KIYPOP
                 ;
BA48 7A          RIGHT: LD    A.D
BA49 FE4A             CP    4AH
BA4B CAF6B9           JP    Z.KIYPOP
BA4E CDCABA           CALL  RSM
BA51 C3F6B9           JP    KIYPOP
                 ;
BA54 7B          LUP:   LD    A.E
BA55 FE00             CP    00H
BA57 CAF6B9           JP    Z.KIYPOP
BA5A 7A               LD    A.D
BA5B FE00             CP    00H
BA5D CAF6B9           JP    Z.KIYPOP
BA60 C5               PUSH  BC
BA61 015000           LD    BC.50H
BA64 A7               AND   A
BA65 ED42             SBC   HL.BC
BA67 C1               POP   BC
BA68 1D               DEC   E
BA69 CD25BB           CALL  LSM
BA6C C3F6B9           JP    KIYPOP
                 ;
BA6F 7B          UP:    LD    A.E
BA70 FE00             CP    00H
BA72 CAF6B9           JP    Z.KIYPOP
BA75 C5               PUSH  BC
BA76 015000           LD    BC.0050H
BA79 A7               AND   A
BA7A ED42             SBC   HL.BC
BA7C C1               POP   BC
BA7D 1D               DEC   E
BA7E C3F6B9           JP    KIYPOP
                 ;
BA81 7B          RUP:   LD    A.E
BA82 FE00             CP    00H
BA84 CAF6B9           JP    Z.KIYPOP
BA87 7A               LD    A.D
BA88 FE4A             CP    4AH
BA8A CAF6B9           JP    Z.KIYPOP
BA8D C5               PUSH  BC
BA8E 015000           LD    BC.0050H
BA91 A7               AND   A
BA92 ED42             SBC   HL.BC
BA94 C1               POP   BC
BA95 1D               DEC   E
BA96 CDCABA           CALL  RSM
BA99 C3F6B9           JP    KIYPOP
                 ;
                 ;
BA9C D5          WAT:   PUSH  DE                  ;ｼﾞｶﾝ ﾏﾁ
BA9D 1602             LD    D.02H
BA9F 1EAF        W2:    LD    E.0AFH
BAA1 1D          W1:    DEC   E
BAA2 C2A1BA           JP    NZ.W1
BAA5 15               DEC   D
BAA6 C29FBA           JP    NZ.W2
BAA9 D1               POP   DE
```

```
BAAA C9              RET
                ;
                ;
BAAB F3         DISP:   DI                          ;ｸﾞﾗﾌ ﾉ ﾋｮｳｼﾞ ﾙｰﾁﾝ ﾜﾘｺﾐ NO
BAAC D35E               OUT  (GRAM2).A              ;ｶﾗｰ ﾉ ｲﾛ ｸﾞﾘｰﾝ
BAAE C5                 PUSH BC
BAAF 0E12               LD   C.12H
BAB1 0606       LOP2:   LD   B.06H
BAB3 1A         LOP1:   LD   A.(DE)
BAB4 77                 LD   (HL).A
BAB5 23                 INC  HL
BAB6 13                 INC  DE
BAB7 05                 DEC  B
BAB8 C2B3BA             JP   NZ.LOP1
BABB C5                 PUSH BC
BABC 014A00             LD   BC.004AH
BABF 09                 ADD  HL.BC
BAC0 C1                 POP  BC
BAC1 0D                 DEC  C
BAC2 C2B1BA             JP   NZ.LOP2
BAC5 C1                 POP  BC
BAC6 D35F               OUT  (MRAM).A               ;ﾒｲﾝ ﾒﾓﾘｰ ﾆ ｽﾙ
BAC8 FB                 EI                          ;ﾜﾘｺﾐ OK
                ;
BAC9 C9                 RET
                ;
BACA E5         RSM:    PUSH HL                     ;RIGHT BIT
BACB 2184BB             LD   HL.SSCD
BACE 7E                 LD   A.(HL)
BACF FE10               CP   16
BAD1 C2F7BA             JP   NZ.RSR1
BAD4 D5                 PUSH DE
BAD5 1608               LD   D.08H
BAD7 2185BB     RSM1:   LD   HL.CDATA1
BADA CD67BB             CALL LRSM
BADD 21F1BB             LD   HL.CDATA2
BAE0 CD67BB             CALL LRSM
BAE3 215DBC             LD   HL.CDATA3
BAE6 CD67BB             CALL LRSM
BAE9 15                 DEC  D
BAEA C2D7BA             JP   NZ.RSM1
BAED D1                 POP  DE
BAEE E1                 POP  HL
BAEF 14                 INC  D
BAF0 23                 INC  HL
BAF1 3E08               LD   A.08H
BAF3 3284BB             LD   (SSCD).A
BAF6 C9                 RET
                ;
BAF7 34         RSR1:   INC  (HL)
BAF8 2185BB             LD   HL.CDATA1
BAFB CD0CBB             CALL RRSM
BAFE 21F1BB             LD   HL.CDATA2
BB01 CD0CBB             CALL RRSM
BB04 215DBC             LD   HL.CDATA3
BB07 CD0CBB             CALL RRSM
BB0A E1                 POP  HL
BB0B C9                 RET
                ;
BB0C C5         RRSM:   PUSH BC
```

```
BB0D 0E12          LD   C,18
BB0F 7E      RBA1: LD   A,(HL)
BB10 CB3F          SRL  A
BB12 77            LD   (HL),A
BB13 23            INC  HL
BB14 0605          LD   B,05H
BB16 7E      RBB1: LD   A,(HL)
BB17 CB1F          RR   A
BB19 77            LD   (HL),A
BB1A 23            INC  HL
BB1B 05            DEC  B
BB1C C216BB        JP   NZ,RBB1
BB1F 0D            DEC  C
BB20 C20FBB        JP   NZ,RBA1
BB23 C1            POP  BC
BB24 C9            RET
             ;
BB25 E5      LSM:  PUSH HL              ;LEFT BIT
BB26 2184BB        LD   HL,SSCD
BB29 7E            LD   A,(HL)
BB2A FE00          CP   00
BB2C C252BB        JP   NZ,LSR1
BB2F D5            PUSH DE
BB30 1608          LD   D,08H
BB32 2185BB  LSM1: LD   HL,CDATA1
BB35 CD0CBB        CALL RRSM
BB38 21F1BB        LD   HL,CDATA2
BB3B CD0CBB        CALL RRSM
BB3E 215DBC        LD   HL,CDATA3
BB41 CD0CBB        CALL RRSM
BB44 15            DEC  D
BB45 C232BB        JP   NZ,LSM1
BB48 D1            POP  DE
BB49 E1            POP  HL
BB4A 2B            DEC  HL
BB4B 15            DEC  D
BB4C 3E08          LD   A,08H
BB4E 3284BB        LD   (SSCD),A
BB51 C9            RET
             ;
BB52 35      LSR1: DEC  (HL)
BB53 2185BB        LD   HL,CDATA1
BB56 CD67BB        CALL LRSM
BB59 21F1BB        LD   HL,CDATA2
BB5C CD67BB        CALL LRSM
BB5F 215DBC        LD   HL,CDATA3
BB62 CD67BB        CALL LRSM
BB65 E1            POP  HL
BB66 C9            RET
             ;
BB67 C5      LRSM: PUSH BC
BB68 016B00        LD   BC,107
BB6B 09            ADD  HL,BC
BB6C 0E12          LD   C,18
BB6E 7E      LBA1: LD   A,(HL)
BB6F CB27          SLA  A
BB71 77            LD   (HL),A
BB72 2B            DEC  HL
BB73 0605          LD   B,05H
BB75 7E      LBB1: LD   A,(HL)
BB76 CB17          RL   A
```

```
BB78 77                 LD     (HL).A
BB79 2B                 DEC    HL
BB7A 05                 DEC    B
BB7B C275BB             JP     NZ.LBB1
BB7E 0D                 DEC    C
BB7F C26EBB             JP     NZ.LBA1
BB82 C1                 POP    BC
BB83 C9                 RET
                 ;
                 ;
BB84             SSCD:  DS     1                        ;ｽｸﾛｰﾙ ﾄﾞｯﾄ ｽｳ
                 ;
                 ;
BB85 00000000    CDATA1:DB     00H.00H.00H.00H.00H.00H
BB89 0000
BB8B 00001800           DB     00H.00H.18H.00H.00H.00H
BB8F 0000
BB91 00FFFFFF           DB     00H.0FFH.0FFH.0FFH.00H.00H
BB95 0000
BB97 00FFFFFF           DB     00H.0FFH.0FFH.0FFH.07H.00H
BB9B 0700
BB9D 00001800           DB     00H.00H.18H.00H.0FH.00H
BBA1 0F00
BBA3 0001FFC0           DB     00H.01H.0FFH.0C0H.1FH.00H
BBA7 1F00
BBA9 00031FE0           DB     00H.03H.1FH.0E0H.3FH.00H
BBAD 3F00
BBAF 000C1FFF           DB     00H.0CH.1FH.0FFH.0FFH.00H
BBB3 FF00
BBB5 00301FFF           DB     00H.30H.1FH.0FFH.0FFH.00H
BBB9 FF00
BBBB 00C01FFF           DB     00H.0C0H.1FH.0FFH.0FEH.00H
BBBF FE00
BBC1 00FFFFE0           DB     00H.0FFH.0FFH.0E0H.00H.00H
BBC5 0000
BBC7 00FFFFC0           DB     00H.0FFH.0FFH.0C0H.00H.00H
BBCB 0000
BBCD 00FFFF80           DB     00H.0FFH.0FFH.80H.00H.00H
BBD1 0000
BBD3 003FFC00           DB     00H.3FH.0FCH.00H.00H.00H
BBD7 0000
BBD9 00030C00           DB     00H.03H.0CH.00H.00H.00H
BBDD 0000
BBDF 00C30C00           DB     00H.0C3H.0CH.00H.00H.00H
BBE3 0000
BBE5 007FFFE0           DB     00H.7FH.0FFH.0E0H.00H.00H
BBE9 0000
BBEB 00000000           DB     00H.00H.00H.00H.00H.00H
BBEF 0000
                 ;
                 ;
                 ;
BBF1 00000000    CDATA2:DB     00H.00H.00H.00H.00H.00H
BBF5 0000
BBF7 00001800           DB     00H.00H.18H.00H.00H.00H
BBFB 0000
BBFD 0000FF00           DB     00H.00H.0FFH.00H.00H.00H
BC01 0000
BC03 0000FF00           DB     00H.00H.0FFH.00H.07H.00H
BC07 0700
BC09 00001800           DB     00H.00H.18H.00H.0FH.00H
```

```
BC0D 0F00
BC0F 0001FFC0          DB    00H,01H,0FFH,0C0H,1FH,00H
BC13 1F00
BC15 00031FE0          DB    00H,03H,1FH,0E0H,3FH,00H
BC19 3F00
BC1B 000C1FFF          DB    00H,0CH,1FH,0FFH,0FFH,00H
BC1F FF00
BC21 00301FFF          DB    00H,30H,1FH,0FFH,0FFH,00H
BC25 FF00
BC27 00C01FFF          DB    00H,0C0H,1FH,0FFH,0FEH,00H
BC2B FE00
BC2D 00FFFFE0          DB    00H,0FFH,0FFH,0E0H,00H,00H
BC31 0000
BC33 00FFFFC0          DB    00H,0FFH,0FFH,0C0H,00H,00H
BC37 0000
BC39 00FFFF80          DB    00H,0FFH,0FFH,80H,00H,00H
BC3D 0000
BC3F 003FFC00          DB    00H,3FH,0FCH,00H,00H,00H
BC43 0000
BC45 00030C00          DB    00H,03H,0CH,00H,00H,00H
BC49 0000
BC4B 00C30C00          DB    00H,0C3H,0CH,00H,00H,00H
BC4F 0000
BC51 007FFFE0          DB    00H,7FH,0FFH,0E0H,00H,00H
BC55 0000
BC57 00000000          DB    00H,00H,00H,00H,00H,00H
BC5B 0000
                 ;
BC5D 00000000 CDATA3:DB    00H,00H,00H,00H,00H,00H
BC61 0000
BC63 00001800          DB    00H,00H,18H,00H,00H,00H
BC67 0000
BC69 00001800          DB    00H,00H,18H,00H,00H,00H
BC6D 0000
BC6F 00001800          DB    00H,00H,18H,00H,07H,00H
BC73 0700
BC75 00001800          DB    00H,00H,18H,00H,0FH,00H
BC79 0F00
BC7B 0001FFC0          DB    00H,01H,0FFH,0C0H,1FH,00H
BC7F 1F00
BC81 00031FE0          DB    00H,03H,1FH,0E0H,3FH,00H
BC85 3F00
BC87 000C1FFF          DB    00H,0CH,1FH,0FFH,0FFH,00H
BC8B FF00
BC8D 00301FFF          DB    00H,30H,1FH,0FFH,0FFH,00H
BC91 FF00
BC93 00C01FFF          DB    00H,0C0H,1FH,0FFH,0FEH,00H
BC97 FE00
BC99 00FFFFE0          DB    00H,0FFH,0FFH,0E0H,00H,00H
BC9D 0000
BC9F 00FFFFC0          DB    00H,0FFH,0FFH,0C0H,00H,00H
BCA3 0000
BCA5 00FFFF80          DB    00H,0FFH,0FFH,80H,00H,00H
BCA9 0000
BCAB 003FFC00          DB    00H,3FH,0FCH,00H,00H,00H
BCAF 0000
BCB1 00030C00          DB    00H,03H,0CH,00H,00H,00H
BCB5 0000
BCB7 00C30C00          DB    00H,0C3H,0CH,00H,00H,00H
BCBB 0000
BCBD 007FFFE0          DB    00H,7FH,0FFH,0E0H,00H,00H
```

```
BCC1 0000
BCC3 00000000         DB    00H,00H,00H,00H,00H,00H
BCC7 0000
               :
               :
BCC9           SPSAVE:DS    02H
BCCB                  DS    60
BD07           STACK:
               :
BD07                  END
```

# マシン語アラカルト

*à la carte 1*

# (　)はどういうときに使うのだろう

①(HL)とHLの違いは?
②(8010H)と8010Hの違いは?

たとえば，ペアレジスタを使用するときに，

```
LD HL, 0F3C8H
LD (HL), A
```

のように(　)がある場合とない場合があります．この2つはどう違うのか，まず考えてみましょう．

## HLと(HL)は違う

ペアレジスタを使用するとき，(　)のあるのとないのでは意味が全く違います．

**●(　)がない場合**

ペアレジスタにデータを入れる

例　LD HL, 0F3C8H

HLペアレジスタにF3C8Hをセットします(HレジスタにはF3H，LレジスタにはC8Hが入ります)．

**●(　)がある場合**

(　)の中のペアレジスタに入っているデータをアドレス・データとして，そのアドレスへデータをロードする

例①　LD HL, 0F3C8H
　②　LD (HL), A

①でHLペアレジスタにF3C8Hがセットされます．②では，HLペアレジスタをアドレスを示すデータとして扱い，HLペアレジスタの指し示す番地へAレジスタのデータを書き込みます．この場合は，F3C8H番地にAレジスタのデータが書き込まれます．

## 直接アドレッシングと間接アドレッシング

また，ただF3C8H番地へAレジスタのデータを書き込むだけならば，次のようにも書くこともできます．

```
LD (0F3C8H), A
```

直接アドレッシング

```
=LD HL, 0F3C8H
 LD (HL), A
```

間接アドレッシング

“LD (0F3C8H), A”は直接アドレスを指定していますので，直接アドレッシングともいいます．それに対して，1回HLペアレジスタなどにアドレスの値をセットしてから，メモリに書き込む方法を間接アドレッシングといいます．それぞれの用途は次のように考えていいでしょう．

## 直接アドレッシングと間接アドレッシングの用途

**●直接アドレッシング**

```
LD (△△△△H), A
```

1バイトや2バイト程度のデータで，決まった番地でなおかつ連続性がないとき

に使用する

**●間接アドレッシング**

LD HL, △△△△H

LD (HL), A

連続性のある番地で，データ数が多いときに使用する

この間接アドレッシングは，HLペアレジスタを使用していますので，HLペアレジスタに1を加えたり，1を引いたり，または他のペアレジスタともたし算引き算ができますので，その分だけ，アドレスを指定するのが楽になります．

---

次に，レジスタでなく数値に（ ）がつく場合を考えてみましょう．たとえば，

LD HL, 8010H

LD HL, (8010H)

の2つはどう違うのでしょうか．形が同じで同じ命令のようにみえますが，実は全く違うのです．

## 8010Hは数値，(8010H)は番地を表す

① LD HL, 8010H

Hレジスタに80H，Lレジスタに10Hをセットする

② LD HL, (8010H)

16ビット・データをメモリから転送する命令．8010H番地のデータをLレジスタにセットし，8011H番地のデータをHレジスタにセットする

①はHLペアレジスタに直接データをセットするのに対し，②は（ ）で指定されるメモリ番地と次の番地にあるデータをHLペアレジスタにセットする命令です（データは逆順にセットされます）．

*à la carte 2*

# マルチタスク可能な割り込み機能

割り込みという機能は一体，どういうものなのだろうか？

## 電話のベルが鳴った

具体的な例をとり説明すると次のようになるでしょう．A君は会社のデスクである仕事をしています．そこに電話のベルがなりました．電話の相手は上司の部長です．その部長から，急いでこの仕事をやってくれとたのまれました．するとA君は今までの仕事を中断して，部長の仕事を始めます．つまり，A君にとっては別の仕事が割り込まれたことになります．A君は部長の仕事を終えると，もとの仕事に戻ります．

しかし，A君は，いつも部長からの仕事をやっていたのでは，本来の自分の仕事ができないのでことわることもできます（実際には無理……）．

## 割り込みを受けたり拒んだり

電話のベルはA君に部長の仕事を割り込ませるきっかけを作っていますが，これはハードウェアでサポートしなければなりません．A君が部長の仕事をしたり，ことわったりするのはソフトウェアでできます．これは，IFFレジスタをコントロールすることによって行います．

「EI」命令は別の仕事を引き受けてもよいですよというものであり，「DI」命令は今は別の仕事を引き受けることができませんという命令です．

**割り込みOK！……EI命令**

IFF（インタラプト・フリップ・フロップ）を0にする

**割り込み不可！……DI命令**

IFFを1にする

PC-8801シリーズでは画面処理にこの割り込みを使用しているので，グラフィックス用のカラープレーンに書き込むときは割り込みを禁止してからカラープレーンに書き込まなくてはなりません．処理が終ったら割り込みOKにします．

*à la carte 3*

# 先入れ後出し スタック・ポインタ

退避命令などに出てくるSP(スタック・ポインタ)の意味と使い方がわからない．

「PUSH，POP，CALL，RET………」の命令を使うときには必ずこのスタック・ポインタ（以後SPと略す）の働きが重要になってきます．このSPについて説明しましょう．

## 1 なぜSPが必要になるのか

まず，どうして，SPが必要なのかを考えてみましょう．Z-80Aの内部レジスタは右図のようになっています．汎用レジスタを中心に話を進めましょう．HLペアレジスタは主に16ビットの演算ができるので，番地の間接指定などに多く使用されます．もし，HLペアレジスタを

VRAMのアドレス指定用に使用していて，さらに，別のアドレスをHLペアレジスタで指定する必要が生じた場合はどうするのでしょう．

## HLペアレジスタがいっぱいなとき

もし，SPを使用できないとしたならば，一番簡単なのはDEペアレジスタやBCペアレジスタなどにEX命令で保存する方法です．また，「EXX命令」で副レジスタセットに保存する方法もあります．

しかし，もし，D，E，B，Cなどのレジスタを他の目的のために使用していたのであれば，この方法はとれません．もし，Aレジスタを使用できるとしたならば，以下のようにHLペアレジスタのデータをメモリに保存することができます．

**●Aレジスタを使ったHLペアレジスタの退避**

```
      ⟆
LD A, H              ┐
LD (0E500H), A       │ HLペアレジスタの保存
LD A, L              │
LD (0E501H), A       ┘
      ⟆              } この間HLペアレジスタは自由に使える
LD A, (0E500H)       ┐
LD H, A              │ HLペアレジスをもとのデータに戻す
LD A, (0E501H)       │
LD L, A              ┘
      ⟆
```

このように，「PUSH，POP命令」を使用しなければ，16バイトのメモリを使用しなくてはなりません．

## SPはレジスタ保存番地の管理人

やっかいな問題は，保存したメモリの番地をプログラマ自身がしっかりと管理しなければならないことです．

これは非常にやっかいなことです．たとえばBC, DE, AFペアレジスタの保存が生じた場合を考えてください．何番地にどのレジスタのデータが保存してあるかを覚えておくのは大変です．これでは，プログラマに負担がかかりすぎます．

そんなときに，レジスタの保存している番地を自動的に管理してくれるのが，SPなのです．このSPのおかげで，プログラマはどの番地にどのレジスタのデータが保存されているかを，全く気にせずに済みます．

## 2 SPはどのように動くのだろう

このSPを使用するにあたっては，専用レジスタのSPにどのメモリエリアを使用するか，アドレスをセットしなければなりません．

セットするには，

```
LD SP, 0E500H
```

とすれば，SP専用レジスタにE500H番地がセットされます．これで，SPが使用可能になります．

## SPとメモリの関係

たとえば「PUSH HL」と「POP HL」命令を使用したときのSPとメモリの関係を図で説明するとP.244の図のようになります．

＊SPとメモリの関係
①現在の状態
メモリ
E4FBH 00
E4FCH 00
E4FDH 00
E4FEH 00
E4FFH 00
E500H 00
SP
E500
SP専用レジスタのデータ
現在この位置を指している
H L
F3 C8
HLレジスタのデータ
②PUSH HL を実行すると…
メモリ
E4FBH 00
E4FCH 00
E4FDH 00
E4FEH C8
E4FFH F3
E500H 00
SP
E4FE
この位置を指す
H L
F3 C8
PUSH HLの実行
(実行後HLレジスタの内容は変らない)
③POP HL を実行すると…
メモリ
E4FBH 00
E4FCH 00
E4FDH 00
E4FEH C8
E4FFH F3
E500H 00
SP
E500
この位置に戻る
H L
F3 C8
POP HLの実行
(実行後メモリの内容は変らない)

このようにSPは番地の小さい方に向かってカウントしているので，一般的にはメモリのワークエリアの途中にはセットせず，終りの方にセットするのが普通です．

## SPエリアの確保のし方

SPの使用メモリのエリアを確保するときには次のようにしなければなりません．

たとえば，30レベルのスタック・エリアを確保するときに必要なバイト数は，

30×2バイト＋1＝61バイト

つまり，連続して30回スタック・エリアを使用する場合は61バイト分必要であることになります．

しかし，実際には，連続して30回データをスタック・ポインタに積むことは，あまりありません．

## 3 他の使用例 CALLとRET

「PUSH，POP」命令ばかりでなく，「CALL，RET」命令でも，SPは使用されています．次の命令のSPの動きを見てみましょう．

```
メインプログラム        サブルーチン
      ∫                      ∫
①CALL WAIT          →WAIT：LD B, 0FH
      ∫                      ∫
                   ②RET
```

①CALL WAITを実行すると，現在のPC（プログラム・カウンタ）のデータをSPで退避します．次にWAITというサブルーチンが入っているアドレスをPCにセットし，実行します．

②RET命令があると，SPで退避していたデータをPCにセットします．PCのデータがもとのデータに戻った，そこの番地から実行を開始します．

なお，このPCは分岐命令などがない限り次の実行すべき番地を指しているので，戻ってもすぐ実行できるのです．

## クリアにも使える

このようにSPは一般的にはレジスタのデータの保存用に使用されますが，他の使い方もあります．SPエリアをVRAMにすれば，これを利用して画面のクリアができます．

また，SPはコンピュータにとって大事な機能なので，この機能を最大限に生かすコンピュータもあります．SPをうまく使用すると，ステップ数が短くなるという特長を利用したもので，逆ポーランド式による計算方法がそれです．

*à la carte 4*

# いろいろ使える スタック操作命令

スタック操作命令にはどんな種類があり，どんなときに使われるのだろうか．

スタック操作命令とはスタック・ポインタを使用する命令のことです．このスタック操作命令は4つに分けて考えることができます．

**スタック・ポインタでデータをセットする命令群（LD 命令）**

| 命令 | 説明 |
|---|---|
| LD SP, 0BFFH | ←SPにデータをセットすることによって，スタック・ポインタを設定することになる |
| LD SP, (nn) | ←nnで指定したアドレスからSPに16ビットデータがセットされる |
| LD (nn), SP | ←nnで指定したアドレスにSPのデータが書き込まれる |
| LD SP, HL | SPに各16ビットレジスタのデータをセットする |
| LD SP, IX | 〃 |
| LD SP, IY | 〃 |
| EX (SP), HL | SPの示すアドレスのデータと各16ビットレジスタのデータを交換する |
| EX (SP), IY | 〃 |
| EX (SP), IX | 〃 |

これらの命令はSPのデータを見たり，新たにセットすることができます．新たにスタック・ポインタの位置をセットする場合や，複数の擬似スタック・ポインタを必要とする場合に活用されます．

**演算命令に関する命令群**

| 命令 | 説明 |
|---|---|
| ADD HL, SP | ←HLペアレジスタとSPとを足し算してHLペアレジスタにセット |
| ADD IX, SP | ←IXインデックス・レジスタとSPとを足し算してHLペアレジスタにセット |
| ADD IY, SP | ←IYインデックス・レジスタとSPとを足し算してHLペアレジスタにセット |
| ADC HL, SP | ←ADD HL, SPと同じだが，キャリーも一緒に加える |
| SBC HL, SP | ←キャリーも含めてHLペアレジスタからSPを引く |
| INC SP | ←SPに1を加える |
| DEC SP | ←SPから1を引く |

## レジスタ退避命令 PUSH, POP

レジスタの退避をする場合に使用され，一番多く使用されます．この退避命令ではPUSHとPOPの順番を間違えないようにしないと大変なことになります．

```
PUSH HL─┐
PUSH DE─┐│   順番を守らない
PUSH BC─┐││   と正しいデータ
PUSH AF┐│││   が戻らない
  ∫
POP AF ←┘
POP BC ←─┘
POP DE ←──┘
POP HL ←───┘
```

スタック・ポインタでデータを保存する場合，初めに退避したデータは一番最後に戻ってきます．

## HLペアレジスタとBCペアレジスタを交換

意図的に順番を変えると面白いこともできます．たとえば，HLペアレジスタとBCペアレジスタのデータを交換したい場合です．残念ながら「EX HL, BC」という命令はないので，この順番を意図的に違えることによって，データを交換しようとするわけです．

**HLペアレジスタとBCペアレジスタの交換**

```
PUSH HL─┐
PUSH BC┐│
POP HL ←┘│
POP BC ←─┘
```

一番最後に「PUSH BC」を使用してスタックにデータを保存し，次の「POP HL」で本来はBCペアレジスタに入るべきデータをHLペアレジスタに入れてしまうのです．この方法は4バイトで済みますので，「LD」命令を使用するよりはるかにプログラムが短くなります．

## 表面に表れない命令群

表面に表れませんがPC（プログラム・カウンタ）の退避を行う命令があります．主に「CALL」，「RET」命令群です．

| | |
|---|---|
| CALL | 無条件にコールする |
| RET | 無条件に戻る |
| CALL Z, nn | Z=1のときコール |
| RET Z | Z=1のとき戻る |
| CALL NZ, nn | Z=0でコール |
| RET NZ | Z=0で戻る |
| CALL C, nn | C=1でコールする |
| RET C | C=1で戻る |
| CALL NCnn | C=0でコールする |
| RET NC | C=0で戻る |
| CALL M, nn | S=1でコールする |
| RET M | S=1で戻る |
| CALL P, nn | S=0でコールする |
| RET P | S=0で戻る |
| CALL PE, nn | P/V=1でコールする |
| RET PE | P/V=1で戻る |
| CALL PO, nn | P/V=0でコールする |
| RET PO | P/V=0で戻る |

（Z, C, S, P/Vはフラグ）

特にこの「CALL」命令を多く使用すると，スタックが積まれますので注意が必要です．

*à la carte 5*

# SPはC000H以前に

マシン語プログラムを作る際，どのようにSPを設定したらよいか．

$N_{88}$-BASICからマシン語のプログラムに入った場合，8レベル（16バイト）分しかマシン語用スタックエリアが用意されていません．少し大きなプログラムになるとこれでは不足してしまい，$N_{88}$-BASICのワークエリアのデータを壊してしまいます．そこで新しくSPを設定しなければなりません．グラフィックスのVRAMのことを考えると，C000H以前にSPを設定しないといけません．そこで，BFF0H番地にSPを設定してみましょう．

```
ORG 0B000H

LD  (0BFFAH), SP  ← N88-BASICのSPの位置をBFFAH番地とBFFBH番地に保存する

LD SP, 0BFF0H     ← マシン語用にBFF0H番地に新しくSPをセットする

    ⟆

LD  (0BFFCH), SP  ← また，マシン語のプログラムを実行させるときのために，BFFCH，BFFDH
                     番地にデータを保存しておく（再び，BASICから入らない場合は必要ない）

LD SP, (0BFFAH)   ← もとのSPの位置に戻しておく必要がある．そうしないと暴走する

RET               ← BASICに戻る
JP MON            ← モニタへ戻る
```

なお，SPをBFF0H番地に設定しているので，BFF0H番地以前の数十バイト分は絶対に使用できません．

また，マシン語のプログラムからBASICのプログラムに戻る場合は，必ずもとのSPの位置に戻してからでないと，暴走をおこします．

*à la carte 6*

# RSTで再開始番地を操作

RST命令がよくわからない．面白い使い方があるのだろうか．

RST命令は再スタート（RESTART）番地を操作する命令です．Z-80Aは外部的に再スタートできる番地を8＋1（NMI）個もっています．その8個の番地は次のようになっています．

**再スタート・アドレス**

```
0000H    (RST 00H)
0008H    (RST 08H)
0010H    (RST 10H)
0018H    (RST 18H)
0020H    (RST 20H)
0028H    (RST 28H)
0030H    (RST 30H)
0038H    (RST 38H)
```

## 割り込み時に参照される再スタート・アドレス

0000H番地はリセットをかけたときにスタートする番地ですから，特別な番地です．他の7個は何故必要なのでしょうか．0008Hから0038Hまでの7個は，ソフトウェア的にどうしても必要なのではありません．本当に必要なのは，Z-80A CPUを使用したシステムで，割り込みをかけるときにどうしても必要な機能です．本来はこの割り込みのときに参照されるアドレスが，この再スタート・アドレスなのです．ソフトウェア的にも使用できるようにRST命令が用意されているのです．PC-8801シリーズでは，モード2という強力な割り込み機能を使用していますので，この再スタート・アドレスは参照していません．そこで，ソフトウェアだけに使用しています．

## モニタに戻るとき

マシン語のプログラムの終りでモニタモードに戻る場合は，本書では次のアドレスにジャンプしていました．

```
JP 0E826H   ←本書で使用
```

この他に，

```
JP 0E669H
```

としても，モニタへ戻ります．実験してみてください．ジャンプ命令を使用すると3バイトを必要とします．

## RSTだと1バイトで済む

しかし，“RST　38H”はモニタのワークエリアとなっているために，RST命令を使用すると，

```
RST  38H   (マシン語でFFH)
```

で，1バイトで済みます．

**モニタモードに戻る3つの方法**

① JP 0E828H　――3バイト命令
② JP 0E669H　――3バイト命令
③ RST 38H　――1バイト命令

## 0038Hには何が書かれているか

このように再スタート・アドレスを参照するようなときは，JP 命令を使用するよりも，1バイトの RST 命令を使用した方が便利です．それでは，0038H 番地には何が書かれているのでしょうか．モニタで逆アセンブルすると右上のリストのようになります．

```
0038    C3 E669     JMP   E669
003B    00          NOP
003C    00          NOP
003D    00          NOP
003E    00          NOP
003F    00          NOP
```

“RST 38H”を実行すると，0038H 番地へジャンプしてここから実行します．すると，0038H 番地には「JP 0E669H」の命令が書いてあります．

```
h]DE000,E0FF
E000 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
E010 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
E020 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
E030 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
E040 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
E050 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
E060 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
E070 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
E080 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00
E090 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00
E0A0 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00
E0B0 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00
E0C0 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00
E0D0 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00
E0E0 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00
E0F0 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00
```

この「RST　38H」は実に面白いことをしてくれます．上のリストは E000H 番地から E0FF 番地までのダンプリストになります．

## RST 38Hで暴走を食い止める

このままの状態で E000H 番地から実行するとどうなるでしょう．

モニタへ戻りましたね．このダンプ上の番地ならばどこからスタートしても，モニタに戻ります．PC-8801シリーズは，電源投入時に使用していないエリアが00H と FFH になっています．

もし，これらの番地にプログラムが暴走してきたならばどこかで CPU は，OP コードとして FFH を読み込み実行します．すると CPU は0038H 番地へ自動的にジャンプしてしまいます．そこはモニタのワークエリアですから，モニタのコマンドとなり，暴走を食い止めることができます．ただし，モニタモードになったからといって，VRAM 上のデータは信用できません．暴走したときにどこの番地を通ってきたかわからないからです．もし，運がよければデータは大丈夫

かもしれません．
各RST命令は，PC-8801シリーズでは次のように使われています．

**RST命令の使用**

| 命　令 | アドレス | 項　目　名 | 機　　能 |
|---|---|---|---|
| RST 00H | 0 0 0 0 | リセット | |
| RST 08H | 0 0 0 8 | シンタックスチェック | RST(or CALL)の後に置かれているキャラクタと(HL)とを比較して，一致しなかったらSyntax error，一致したらRST 10Hをして戻る |
| RST 10H | 0 0 1 0 | テキストからの1文字読み込み | テキストから1文字または数を読み込む |
| RST 18H | 0 0 1 8 | 1文字出力 | CRTまたはLPTに1文字出力をする．Aレジスタに文字コードを入れてRST 18Hを行う<br>(E64C) ＝0 CRTに出力<br>≠0 LPTに出力 |
| RST 20H | 0 0 2 0 | HL，DEの比較 | HL-DE（無符号）を行いフラグをセットする．Aレジスタはこわれる |
| RST 28H | 0 0 2 8 | SGN (FAC) | FAC(単精度，倍精度)の符号を調べる<br>－：Aレジスタ＝FF，NZ，M<br>0：Aレジスタ＝0，Z，P<br>＋：Aレジスタ＝1，NZ，P |
| RST 30H | 0 0 3 0 | FACの型チェック | FACの型を調べる<br>INT：Aレジスタ＝FF，C，NZ，M，PE<br>STR：Aレジスタ＝00，C，Z，P，PE<br>SNG：Aレジスタ＝01，C，NZ，P，PO<br>DBL：Aレジスタ＝05，NC，NZ，P，PE |
| RST 38H | 0 0 3 8 | ユーザ用RST | モニタではブレイク・ポイント用またはコマンド待ちに戻るときに使う |

*à la carte 7*

# DMAを止めてしまうと…

グラフィックスの場合は，DMA処理を止めると，何かと便利だと聞くけど．

## データ転送が速くできる

DMA (Direct Memory Access) とはダイレクト・メモリ・アクセスの略で，通常使用されているCPUを使用しないでメモリとメモリ，メモリとI/Oの間で直接データのやりとりを行う方法です．DMA処理を行うためには，それ専用のハードウェアが必要となりますが，データの転送がCPUを使用するより速くできるのが強みです．

## グラフィックスではDMAはいらない

PC-8801シリーズではDMAは主にCRTCとの間に使用されています．CRTCは定期的にVRAMのデータを読み込むのに使用されておりDMA処理を行っている間はCPUはSTOPしてしまいますので，CPUの見かけの処理速度が遅くなります．

このDMAはテキストモードのときだけ必要です．グラフィックスVRAMだけ使用しているのであればDMA処理は必要ありません．

そこで，DMAを止めてしまうと，30%ぐらいスピードアップできます．テキストモードで使用してもかまいませんが，表示が消えてしまいます．ですから単なる計算のみを速く処理したいのであれば非常に有効な方法です．

## DMAストップでスピードをアップする

マシン語のプログラムではP.253のような命令を入れておくとDMAのストップとスタートが簡単にできます．本書のプログラムにこのプログラムを追加するとスピードが速くなるのがわかります．

## BASICプログラムで使うには

また，BASICプログラムでも使用できます．ただし，テキスト文字は消えますのでDMAをスタートさせてからテキスト文字を表示させるようにすれば楽にできます．

**● DMAのストップ**

```
OUT81, 0または　OUT　104, 0
```

**● DMAのスタート**

```
WIDTH80, 25
```

このプログラムで実行にかかる時間は約30秒です．しかし，20行を消して実行すると約45秒かかります．15秒間は無駄な時間を消費していることがわかります．プログラムを工夫して，どんどん使いましょう．

**●DMAのストップ&スタート**

```
ORG     START

LD      (SPSAVE),SP  ┐
LD      SP,STACK     │
                     │
LD      BC,5019H     │
LD      A,01H        │
LD      (0E6B2H),A   │
LD      A,19H        │
LD      (0E6B3H),A   ├ テキスト画面の設定
LD      A,0E8H       │
LD      (0E6B4H),A   │
LD      A,00H        │
LD      (0E6B8H),A   │
LD      A,0FFH       │
LD      (0E6B9H),A   │
CALL    COLOR        ┘

LD      A,00H        ┐ DMAをストップする
OUT     (51H),A      ┘
        :            ┐
        :            ├ 必要なプログラム
        :            ┘
LD      BC,5019H     ┐
LD      A,01H        │
LD      (0E6B2H),A   │
LD      A,19H        │
LD      (0E6B3H),A   │ DMAをスタートさせる
LD      A,0E8H       ├ (これを忘れるとモニタに戻った
LD      (0E6B4H),A   │  とき画面に何も表示されない)
LD      A,00H        │
LD      (0E6B8H),A   │
LD      A,0FFH       │
LD      (0E6B9H),A   │
CALL    COLOR        ┘

LD      SP,(SPSAVE)
JP      MON          ←── モニタに戻る
END
```

**●BASICでの例**

```
10 REM ******* DMA STOP DEMO PROGRAM ******
20 OUT 81,0
30 FOR N=1 TO 5000
40        A=.1
50        B=B+A
60 NEXT N
70 WIDTH 80,25
80 PRINT"B= ";B
90 END
```

à la carte 8

# タンスのような バンク切り替え

わかったようでよくわからないのが，バンク切り替え．単純明快に説明すると…．

## 64Kバイトでは足りないメモリ

バンク切り替えというのは簡単に言えばメモリの容量を増やす方法のひとつです．PC-8801mkIISRに使用されているCPUは8ビット・CPUのZ-80Aですから，これで一度にメモリをアクセスできるのはFFFFH番地までの64Kバイトです．

一昔前に比べると64Kバイトは決して小さくないのですが，BASICインタプリタのプログラムの大きさが32Kバイト以上もあると，テキストエリアが小さくなってしまいます．

また，グラフィックスのVRAMなどは，48Kバイトあり，とても64Kバイトでは不足してしまいます．

## OUT, IN命令でメモリを増やす

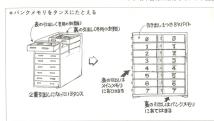
そこで，メモリの大きさを増やすために「OUT，IN」命令をうまく利用しています．「OUT，IN」命令は実行するにはメインメモリと直接関係ないデバイス番号（00H～FFHまで）を指定する必要があります．つまり，デバイス番号のI/O番地にメモリをつけておくと，メインメモリとデバイス番号につけたメモリを交換することが可能です．この方法を利用すればいくらでもメモリを大きくできます．しかし，一度にアクセスできるメモリの大きさは，64Kバイトしかありません．

メモリをタンスにたとえて説明するとP.255のようになります．

## バンクメモリはタンス

二重引出しのタンスを例にとって考えましょう．今は夏の季節としましょう．すると冬用の衣類はほとんど着ないので，奥のタンスの引出しにしまっておきます．

普段は表の引出しを利用していれば間に合います．しかし，非常に冷え込んで冬用のトレーナーなどを引き出したい場合があります．しかし，冬用の引出しは裏の引出しにしまってあるので，すぐにトレーナーを出すことはできません．

そこで，一度表の引出しを引き抜いて裏の引出しを引き出します．すると，表の引出しはタンス上にはなくなります．その代りに裏の引出しが表の引出しの位置に来ます．トレーナーを出すと，裏の引出しは奥においやられて，空いた位置に再び表の引出しが入ります．つまり，表の引出しはメインメモリ，裏の引出しはバンクメモリということになります．

表の引出しと裏の引出しを入れ替えることをバンク切り替えといいます．この例は二重の例ですが，実際には何重にもなります．このバンク切り替えをする命令が「OUT，IN」命令なのです．

*à la carte 9*

# バンク切り替えに似たウィンドウ機能

テキストRAMの内容を読み出すときなどに使うウィンドウ機能とは何だろう．

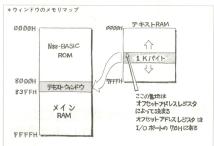
## ROMと重なったTEXTエリアはそのままでは読み出せない

ウィンドウ機能は$N_{88}$-BASICモード（モード1）での特殊な機能ですが，一種のバンク切り替えです．バンクの大きさが1Kバイト単位というだけです．

$N_{88}$-BASICモードでは，CPUが0000H番地から7FFFH番地までのメモリを読み出すときはROMから読み出し，書き込むときは，0000H番地から7FFFH番地までのテキストRAMに書き込みます．すると，テキストRAMの内容を読み出すことができないので，テキストRAMの内容をROMと関係ない8000H番地から83FFH番地に一旦移してから読み出します．ただし，これはソフトウェアでサポートしているのではなく，ハードウェアで自動的に8000H番地からに書き込んでくれるのです．ですから，この8000H番地から83FFH番地に書き込めば，自動的にテキストRAMにも書き込んでくれます．

## オフセットアドレスレジスタで任意に読み書き

しかし，これでは1Kバイトしか読み書きができないので，これを0000H番地から7FFFH番地のどこででもできるようにI/Oポートにオフセットアドレスレジスタをもっています．

このオフセットアドレスレジスタ（上位7ビットレジスタ）で任意の上位バイトのアドレスをセットすることによって，自由に読み書きができます．

オフセットアドレスレジスタはI/Oポートの70Hにあります．この70Hにアドレスの上位バイトをセットすれば良いのです．オフセットアドレスレジスタに関係する命令には次の3つがあります．

①オフセットアドレスレジスタに読み出したいアドレスの上位バイトをセット

OUT（70H），上位バイト

②オフセットアドレスレジスタにセットされている上位バイトを読み出す

IN（70H）

③オフセットアドレスレジスタに1を加える（出力データは何でもよい）

OUT（78H），ダミーデータ

このウィンドウ機能をテストするにはモニタコマンドのD，I，Oコマンドを使用すれば簡単にできます．

## 1000H番地から1バイトをウィンドウに表示

もし，1000H番地から1Kバイトをウィンドウに表示させるのであれば，

h］O70，10↵

とすれば，ウィンドウには1000H番地から13FFH番地まで表示されているはずです．Dコマンドで確めてください．また，

+100H番地からならば，

```
[h 078, 00 ↵
        └このデータは何でもよい
```

とすれば，オフセットアドレスレジスタに1が加えられて11Hとなり，1100H番地から14FFH番地まで表示されることになります．

## 表示されているメモリアドレスを知りたい

ウィンドウに表示されているメモリ番地を知りたいときは，次のようにします．

```
h] I 70↵
23
h]
```

23Hというデータが返ってきたならば，2300H番地から26FFHまでということになります．つまり，返ってきたデータは上位バイトなので，下位バイトに00Hをつけてそこから1Kバイト分ということになります．表示しているアドレスは，

```
  ┌─つけ加える ┌─1Kbite分
2300H  +  3FFH  =  26FFH
```

ウィンドウに表示されている実際の番地の始まり　　ウィンドウに表示されている実際の番地の終り

とすれば簡単に求めることができます．

## どうして面倒な窓を作ったのか

どうして，このような理解しにくい機能をとり入れたのでしょうか．その理由は，BASIC命令にグラフィックス命令などを追加したため，BASICのインタプリタプログラムが大きくなってしまったからです（約32Kバイト．このもとになったN-BASICインタプリタは24Kバイト）．

そのためにテキストRAMが不足したので，バンク切り替えでテキストRAMを増やさなければならなかったのです．そこで単なるバンク切り替えでテキストRAMを増やすと，BASICインタプリタは頻繁にテキストRAMをアクセスするので，実行速度が低下します．そこで1KバイトのテキストRAMをメインメモリに持ち，できるだけバンク切り替えをしなくてすむようにしたわけです．

つまり，1Kバイト以内であるならばバンク切り替えを必要としないのです．1Kバイトより大きくても，オフセットアドレスレジスタに上位バイトのアドレスをセットするだけなので，1回のバンク切り替えで済みます（普通にバンク切り替えをすると，もとのメインメモリに戻るためにはもう1回必要です）．

以上の理由からウィンドウ機能という方法を採用したのだと思われます．

*à la carte 10*

# メモリモードを切り替えるには

メモリモードを切り替えれば，ウィンドウを使わずに直接RAMに書き込める．

PC-8801mkIISRは3つのメモリモードを持っています．

**モード0**　N-BASICモード
**モード1**　$N_{88}$-BASICモード
**モード2**　64K RAMモード

以上の3つのモードをプログラムで切り替えることができます．一番多く使用されるモード1とモード2の切り替えについて説明しましょう．

## どんなときにモード切り替えが必要か

モード切り替えは次のようなときに必要となります．

**①64K RAMモードでプログラムを動かす場合**

64K RAMモードで使用する場合は最初にメモリモードを切り替えて使用する場合が多いようです．

このモードで多く使用されているのは，他のDOS（たとえば，CP/MやUCSD）を使用する場合です．

**②テキストRAMをウィンドウを通さずにアクセスしたい場合**

ウィンドウは1Kバイトなので多量のテキストエリアのデータをアクセスできません．そこで，すばやく処理するために一時的にRAMモードにしてからデータを転送したい場合などに使用されます．ゲームソフトに多いようです．

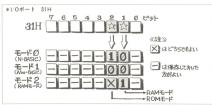
**③64K RAMモードで動いているプログラムから，$N_{88}$-BASICのROM内ルーチンを呼びたい場合**

計算にROM内ルーチンを使用して，プログラムを簡単にしたい場合などに使われます．この場合，BASICのワークエリアをそのまま持っていなければなりません．

## モードを切り替えるには

メモリモードをコントロールしているのは，I/Oポートの31Hの1ビット目と2ビット目です．このビットをコントロールすればいいのです．

モード変更するには，2ビットだけでよいのですが，他のビットは他の用途に使用されているために，勝手に変えることはできません．そこで，「OR，AND」命令を使用します．現在のI/Oポートの31HのデータはE6C2H（$N_{88}$-BASICモード時）に保存されていますので，このデータを利用すればいいことになります．モード1，2にするにはP.259のようにします．

バンク切り替えを行うときは，必ず割り込みを禁止しなければなりません．64K RAM実行中は割り込みを禁止しな

モード2にするとき (N88-BASIC→64K RAM)

```
DI              ──→割り込み禁止
LD  A, (0E6C2H) ──→現在のデータをもってくる
OR  02H         ──→現在のデータの1ビット目を1にする (RAMモード)
OUT (31H), A    ──→バンク切り替えをする
LD  (0E6C2H), A ──→新しいデータを保存する
EI              ──→割り込み許可
```

モード1にするとき (64K RAM→N88-BASIC)

```
DI
LD  A, (0E6C2H)
AND 0F9H        ──→1ビット目と2ビット目を0にして, 他のビットはマスクする
OUT (31H), A
LD  (0F6C2H)
EI
```

いと, N88-BASIC ROMのルーチンを使用する関係で暴走をおこします.

## 64K RAMを使ったサンプルプログラム

原理がわかったところで簡単なサンプルプログラムを作ってみましょう. N88-BASICモードでは,ウィンドウを使用することによって, 0000Hから始まるメモリにデータを書き込むことが簡単にできます.

つまり, モニタモードでSコマンドなどを使用して, 0000H番地に書き込めば, 自動的にRAM側に書き込まれるのです. したがって, 0000H番地に簡単なプログラムをサブルーチンとして書き込

みます．

64K RAMモードにして，サブルーチンをコールしてみましょう．64K RAMモードにした場合はモニタも消えてしまうので，何らかのプログラムを入れておかないと暴走してしまうからです．

プログラムは次のようにします．

**●メインプログラム**

・$N_{88}$-BASICモード(モード1)で64K RAMモード(モード2)に切り替える

・0000H番地からのサブルーチンをコールする

・64K RAMモード（モード2）から$N_{88}$-BASICモードに切り替える

・モニタに戻る

**●サブルーチン**

・0010H番地から，順次，AAHというデータをFFH回書き込む

**●メモリモードの切り替え　その1**

```
                    ;*******************************************
                    ;
                    ; ﾒﾓﾘ ﾓｰﾄﾞ ﾉ ｷﾘｶｴ (ﾓｰﾄﾞ1ｶﾗ 2)
                    ;
                    ; PROGRAM NAME ---> MMTEST2.e
                    ;
                    ;*******************************************
                    ;
B900                START: EQU  0B900H
                    ;
                           ORG  START
                    ;
B900 F3                    DI                     ;ﾜﾘｺﾐ ｷﾝｼ
B901 3AC2E6                LD   A,(0E6C2H)        ; MEMORY MODE
B904 F602                  OR   02H
B906 D331                  OUT  (31H),A
B908 32C2E6                LD   (0E6C2H),A
                    ;
B90B CD0000                CALL 0000H
                    ;
B90E 3AC2E6                LD   A,(0E6C2H)
B911 E6F9                  AND  0F9H
B913 D331                  OUT  (31H),A
B915 32C2E6                LD   (0E6C2H),A
B918 FB                    EI
B919 FF                    RST  38H               ;ﾓﾆﾀｰﾆ ﾓﾄﾞﾙ
B91A                       END
```

**●ウィンドウに表れているプログラムをモニタを使って逆アセンブルした**

```
8000    21 0010    LXI   H,0010
8003    3E AA      MVI   A,AA
8005    06 FF      MVI   B,FF
8007    77         MOV   M,A
8008    23         INX   H
8009    05         DCR   B
800A    C2 0007    JNZ   0007
800D    C9         RET
```

ウィンドウを使用して，メインプログラムを実行させる前に，0000H番地から書き込まなくてはなりません．ウィンドウの関係で番地が8000H番地からになっていますが，実際には0000H番地から書き込まれています．サブルーチンはインテルニーモニックになっていますが，これは，モニタのLコマンドで逆アセンブルしたためです．

これを書き込んだ後にプログラムを実行すると，すぐにモニタに戻ります．そこで，Eコマンドで8000H番地からのウィンドウを見てください．しっかりと実行されていることがわかります．ここではメインプログラムを説明するために簡単にし，サブルーチンをメインプログラム中に入れていませんが，応用する場合にはCALL命令を実行する前に，サブルーチンの転送プログラムを入れておく必要があります．

0000H番地のサブルーチンをメインプログラムに入れたプログラムは次のようになります．これで，ウィンドウを使用せずRAM全体に書き込み，読み込みができるようになりましたね．これを少し応用するとG-VRAMをデータ保存のRAMにしたりすることが可能です．

**●メモリモードの切り替え　その2**

```
                 ;****************************************
                 ;
                 ; ﾒﾓﾘ ﾓｰﾄﾞ ﾉ ｷﾘｶｴ (ﾓｰﾄﾞ1ｶﾗ 2)
                 ;
                 ; PROGRAM NAME ---> MB1.e  (WRITE & SUBG0)
                 ;
                 ;****************************************
                 ;
B900             START: EQU  0B900H
                 ;
                 ;
                        ORG  START
                 ;
B900 F3                 DI                          ;ﾜﾘｺﾐ ｷﾝｼ
B901 3AC2E6             LD   A,(0E6C2H)             ; MEMORY MODE 2
B904 F602               OR   02H
B906 D331               OUT  (31H),A
B908 32C2E6             LD   (0E6C2H),A
                 ;
B90B 2125B9             LD   HL,SSUB                ;0000H ﾆ ｻﾌﾞﾙｰﾁﾝ ｦ ｶｷｺﾑ
B90E 110000             LD   DE,0000H
B911 010F00             LD   BC,000FH
B914 EDB0               LDIR
                 ;
B916 CD0000             CALL 0000H
                 ;
B919 3AC2E6             LD   A,(0E6C2H)             ; MEMORY MODE 1
B91C E6F9               AND  0F9H
B91E D331               OUT  (31H),A
B920 32C2E6             LD   (0E6C2H),A
                 ;
B923 FB                 EI
B924 FF                 RST  38H                    ;ﾓﾆﾀｰﾆ ﾓﾄﾞﾙ
                 ;
                 ;
B925 211000      SSUB:  DB   21H,10H,00H            ;LD HL,0010H
B928 3EAA               DB   3EH,0AAH               ;LD A,AAH
B92A 06FF               DB   06H,0FFH               ;LD B,FFH
B92C 77                 DB   77H                    ;LOOP:LD (HL),A
B92D 23                 DB   23H                    ;INC HL
B92E 05                 DB   05H                    ;DEC B
B92F C20700             DB   0C2H,07H,00H           ;JP NZ,LOOP
B932 C9                 DB   0C9H                   ;RET
B933 0000               DB   00H,00H
                 ;
B935                    END
```

●サブルーチンと実行結果

```
8000 21 10 00 3E AA 06 FF 77 23 05 C2 07 00 C9 6F 61  !  >ｪ #  ﾂ  ﾉ
8010 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA  ｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪ
8020 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA  ｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪ
8030 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA  ｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪ
8040 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA  ｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪ
8050 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA  ｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪ
8060 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA  ｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪ
8070 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA  ｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪ
8080 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA  ｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪ
8090 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA  ｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪ
80A0 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA  ｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪ
80B0 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA  ｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪ
80C0 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA  ｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪ
80D0 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA  ｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪ
80E0 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA  ｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪ
80F0 AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA  ｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪｪ
```



注：ウィンドウを使用しているので，8000H番地からになっているが，実際は0000H番地

*à la carte 11*

# NEWしたプログラムを復活するには

BASICプログラムを間違えて消してしまったとき，何とか復活できないか？

間違ってNEW命令を実行したり，STOPキーを押さないで，RESETキーを押してしまうと，BASICのプログラムが消えてしまいます．こんなときにプログラムを復活する方法はないのでしょうか．マシン語を利用すれば復活が可能です．

## リンクポインタのみを復活すればよい

NEW命令やRESETをかけるとメモリにすべて00Hが書き込まれるわけでなく，リンクポインタといわれる部分と，終りを示すポインタのみが消されるだけなのです．プログラムを復活するにはリンクポインタのみを復活すればよいのです．

DISK-BASICについてくる「dskut1.n88」のプログラムを例にとって説明してみましょう．NEW命令で消されるのはテキストエリアの最初のリンクポインタが00Hになるだけですから，最初のリンクポインタを見つけて，正しいデータをセットするだけです．

一般的に初めの1行はP.263のような構成をしています．

**＊リンクポインタの１行目**

| 例 | 10行 | 0001 | 0700 | 0A00 | 8F | 00 |
|---|---|---|---|---|---|---|
| | | ①格納アドレス | ②リンクポインタ | ③行番号 | ④RAMの中間言語 | ⑤行の終りの目印 |

**①格納アドレス**

新しいプログラムはすべて，テキストエリアの0001H番地から格納されます．

**②リンクポインタ**

次の行が格納されている番地．Z-80 CPUの関係上，上位バイトと下位バイトが入れ替わっています．このリンクポインタのデータが0000Hの場合は，特別な意味があります．つまり，

0000H…BASICプログラムの終り

を指し示します．したがって0000H以外のデータならば，次の行があるということになります．

**③行番号**

BASICで使用されている行番号．16進数の4桁なので，BASICで使用できる行番号は最高で65535行です．この例では0A00Hの上位・下位バイトを入れ替えると000AHなのでBASICの行番号は10行．

**④ BASICの中間言語**

BASICコマンドはそのままストア(格納)されるのではなく，１バイトから２バイトの中間言語に変換されてからストアされます．中間言語がわかれば，マシン語でBASICのプログラムの書き替えや解析も可能となります．

**⑤行の終りの目印**

この00Hは行の終りを示すデータです．ここの00Hは次のリンクポインタを見つける際の唯一の手がかりとなっています．

## プログラムの復活

「dskut1．n88」のプログラムを使用してみましょう．実際にメモリがどのようにストアされているか，ウィンドウを通してテキストエリアの0000H番地を表示させると，次のようになります．

**●テキストエリアの0000H番地**

```
hJD8000,80FF
8000 00 07 00 E8 03 8F 00 40 00 F2 03 8F 20 20 20 20    ♠ + @ 年 +
8010 20 44 49 53 4B 20 55 54 49 4C 49 54 59 20 66 6F   DISK UTILITY fo
8020 72 20 50 43 2D 38 38 30 31 4D 4B 32 20 28 20 31  r PC-8801MK2 ( 1
8030 20 64 72 69 76 65 20 73 79 73 74 65 6D 20 29 00   drive system )
8040 46 00 FC 03 8F 00 80 00 06 04 8F 20 20 20 20 43  F   + _   +    C
8050 6F 70 79 72 69 67 68 74 20 28 43 29 20 31 39 38  opyright (C) 198
8060 33 20 62 79 20 50 65 72 73 6F 6E 61 6C 20 43 6F  3 by Personal Co
8070 6D 70 75 74 65 72 20 44 69 76 2C 20 4E 45 43 00  mputer Div, NEC
8080 86 00 10 04 8F 00 B1 00 1A 04 8F 20 20 20 20 20  ■   + ア   +
8090 20 20 20 20 20 20 20 20 20 56 65 72 73 69 6F 6E           Version
80A0 20 31 2E 30 20 20 28 38 33 2F 30 39 2F 32 32 29   1.0  (83/09/22)
80B0 00 B7 00 24 04 8F 00 EA 00 2E 04 20 20 92 20 2C   ｷ $ + ◆ .  ｲ ,
80C0 0C FF E5 20 3A 8F E9 20 63 68 65 63 6B 20 45 78   ◣ :+♥ check Ex
80D0 70 61 6E 64 65 64 20 53 74 61 74 65 6D 65 6E 74  panded Statement
80E0 20 50 61 63 6B 61 67 65 2E 00 F7 00 38 04 20 20   Package. 秒 8
80F0 92 20 2C 0C FF CF 00 03 01 42 04 20 20 AB 20 41  ｲ , ﾏ   B   ｵ A
```

現在のウィンドウは0000H番地から表示している

この中で大切なのは1行目です．この行にプログラムの復活の秘密が隠されています．その秘密をこれから説明していきましょう．

初めのリンクポインタ0700Hに注目してください．このプログラムをNEW命令で消してしまうと1行目はどうなるでしょうか．

初めのリンクポインタに0000Hが書き込まれる以外は，すべてメモリ上に残っています．ここに次のリンクポインタのデータを書き込めば，プログラムは復活するはずです．

## 復活の手がかりは00H

リンクポインタの復活のためには次のリンクポインタを探さなければなりません．それには行の終りの目印である00Hを探せばよいことになります．ただし，復活させようとするリンクポインタの次の2バイトは行番号ですから，どちらかに00Hが入っていることもありますので，4バイト目以降の00Hを探します．00Hを探していくと，8006H番地に00Hがあります．すると次の番地からリンクポインタが始まります．次の番地は8007H番地ですが，復活しようとするリンク

ポインタに0780Hと書き込んではいけません．それはウィンドウを利用しているので8000H番地代になっていますが，本当のメモリは0000H番地代から始まっているからです．したがって“0700H”と書き込まなくてはいけません．

これでリンクポインタが復活しました．BASICモードに戻ってLISTをとってください．BASICプログラムが復活してるはずです．

初めのリンクポインタに0000Hが書かれていると，NEW命令を実行したときと同じになります．また違う位置のリンクポインタのところに0000Hが書かれているとプログラムの終りを意味します．したがって，プログラムの終りを探すには00Hが3バイト書き込まれている番地を探せばよいのです．

## BASICプログラム復活の手順

8000H番地からのデータは，0000H番地から始まるテキストエリアとは限りません．そこで，どうしてもウィンドウが表示している番地を0000H番地にしなければなりません．ウィンドウを0000H番地にするには，

h] 070, 0 ↵

ウィンドウの表示番地を0000H番地からにする

この命令を実行することによって，ウィンドウが0000H番地になります．

**復活の手順**

①MON↵…モニタモードにする

②h] 070, 0↵…ウィンドウを0000H番地にする

③h] e8000↵…ウィンドウをエディットモードにする

④次のリンクポインタの位置を探す（00Hを手がかりにする．00Hの次の位置が次のリンクポインタ）

⑤上位バイトと下位バイトを入れ替えてリンクポインタに書き込む

⑥エディットモードを終了して，BASICモードに戻る

⑦確認のため，LISTをとる

⑧終り

## 簡単なプログラムで復活の練習

プログラムの復活の方法は大事なことなので，簡単なプログラムを使って練習してみましょう．まず“NEW↵”でプログラムを消します．

モニタでダンプリストをとってみます（P.266参照）．

**●消すBASICプログラム**

```
10 REM ***** TEST PROGRAM ****
20 PRINT " *** RETEST PROGRAM ****"
30 END
```

**●ダンプリスト**

```
8000 00 1F 00 0A 00 8F 20 2A 2A 2A 2A 2A 20 54 45 53     + ***** TES
8010 54 20 50 52 4F 47 52 41 4D 20 2A 2A 2A 2A 00 40  T PROGRAM **** @
8020 00 14 00 91 20 22 20 2A 2A 2A 20 52 45 54 45 53    ┬ " *** RETES
8030 54 20 50 52 4F 47 52 41 4D 20 2A 2A 2A 2A 22 00  T PROGRAM ****"
8040 46 00 1E 00 81 00 00 00 1E 04 8F 20 20 20 20 43  F   _     +    C
8050 6F 70 79 72 69 67 68 74 20 28 43 29 20 31 39 38  opyright (C) 198
8060 33 20 62 79 20 50 65 72 73 6F 6E 61 6C 20 43 6F  3 by Personal Co
8070 6D 70 75 74 65 72 20 44 69 76 2C 20 4E 45 43 00  mputer Div, NEC
8080 86 00 10 04 8F 00 B1 00 1A 04 8F 20 20 20 20 20  ■   + ア   +
8090 20 20 20 20 20 20 20 20 20 56 65 72 73 69 6F 6E           Version
80A0 20 31 2E 30 20 20 28 38 33 2F 30 39 2F 32 32 29   1.0  (83/09/22)
80B0 00 B7 00 24 04 8F 00 EA 00 2E 04 20 20 92 20 2C   ｷ $ + ◆ .   ┤ ,
80C0 0C FF E5 20 3A 8F E9 20 63 68 65 63 6B 20 45 78    ◣ :+♥ check Ex
80D0 70 61 6E 64 65 64 20 53 74 61 74 65 6D 65 6E 74  panded Statement
80E0 20 50 61 63 6B 61 67 65 2E 00 F7 00 38 04 20 20   Package. 秒 8
80F0 92 20 2C 0C FF CF 00 03 01 42 04 20 20 AB 20 41  ┤ ,  ﾏ   B   ｵ A
```

次の問題を解いてみましょう．

問1　初めのリンクポインタが0000Hとなっています．ここに書き込む次のリンクポインタの位置を探してください．

（ヒント）　初めのリンクポインタの位置から4バイト以降の00Hを探すこと．

問2　このプログラムを復活させてください．

問3　このプログラムは何番地で終っているでしょうか．

## プログラム復活の解答

**問1**

00Hのある位置は801EH番地なので，次の番地801FH番地となります．したがって，メモリ上の番地は001FH番地となります．

**問2**

次のリンクポインタの位置が001FHなので，

```
8000  00  00  00  0A  00……
          ↓   ↓
          1F  00
```

8001番地に1FHを書き込むだけでよいのです．

BASICモードに戻ればリストがとれます．

**問3**

00Hが3個連続している位置を探すか，最後のリンクポインタのデータをみればよいのです．答は0046Hとなります．

*à la carte 12*

# 自由にファンクションキーを設定したい

BASICのKEY命令のように，マシン語でも自由に設定できないか．

マシン語でも，BASICの「KEY」命令のように自由にファンクションキーを設定できたら，何かと便利です．比較的楽にしようと思ったら，ROM内ルーチンを呼びます．

## ファンクションキー・データのあるアドレス

ファンクションキーのデータは次のアドレスにあります．

| | | |
|---|---|---|
| F.1 | E6F2H〜E701H | 16バイト |
| F.2 | E702H〜E711H | 16バイト |
| F.3 | E712H〜E721H | 16バイト |
| F.4 | E722H〜E731H | 16バイト |
| F.5 | E732H〜E741H | 16バイト |
| F.6 | E742H〜E751H | 16バイト |
| F.7 | E752H〜E761H | 16バイト |
| F.8 | E762H〜E771H | 16バイト |
| F.9 | E772H〜E781H | 16バイト |
| F.10 | E782H〜E791H | 16バイト |

各キーはそれぞれ16バイトということになっていますが，次のファンクションキーのデータと区別するために00Hが必要なので，実際に定義できる文字は15文字までになります．CRT上に出る1つのファンクションキーの内容表示は，11文字ですから充分な大きさでしょう．もし表示される文字が11文字でもよいからもう少し長くしたい場合は，文字列の最後に00Hを入れなければ次のファンクションキーのデータとつながります．この場合は，31文字として扱われます．

## ファンクションキーのデータを変える手順

①ファンクションキーのデータエリアにASCIIコードで任意の文字を書く（モニタを使ってもよい）
②E6B8HにFFHを書き込む
③ROM内ルーチンの「3F7AH」をコール
④終了

実際に実行してみましょう．モニタのDコマンドは，16バイト以上はスタート・アドレスとエンド・アドレスを指定しなければならないので，手間がかかります．そこで，いつも決まっている番地をファンクションキーに設定しておけば楽になります．本書の一部で使用している番地を設定してみましょう．マシン語のスタート・アドレスはB900H番地です．

ファンクションキーに次のように割り当てます．

F1──→ DB900, BFFF
F2──→ LB900, BFFF
F3──→ AB900
F4──→ MB900, BFFF, D900
F5──→ GB900 C/R

これらのデータをブロック転送命令の「LDIR」命令でデータ転送しています.

プログラムは次のようになります.

**●ファンクションキーのコントロール**

```
                 ;*******************************************
                 ;
                 ; ファンクションキー ノ コントロール
                 ;
                 ; PROGRAM NAME --->FC1.e
                 ;
                 ;*******************************************
                 ;
B900             START: EQU  0B900H
BFFF             DEND:  EQU  0BFFFH
3F79             FCSUB: EQU  3F79H
E6F2             FK1:   EQU  0E6F2H
E702             FK2:   EQU  0E702H
E712             FK3:   EQU  0E712H
E722             FK4:   EQU  0E722H
E732             FK5:   EQU  0E732H
                 ;
                        ORG  START
                 ;
B900 3EFF               LD   A,0FFH           ;F.KEY ON
B902 32B8E6             LD   (0E6B8H),A
                 ;
B905 2140B9             LD   HL,FCHR1
B908 11F2E6             LD   DE,FK1
B90B 010A00             LD   BC,000AH
B90E EDB0               LDIR
                 ;
B910 214AB9             LD   HL,FCHR2
B913 1102E7             LD   DE,FK2
B916 010A00             LD   BC,000AH
B919 EDB0               LDIR
                 ;
B91B 2154B9             LD   HL,FCHR3
B91E 1112E7             LD   DE,FK3
B921 010500             LD   BC,0005H
B924 EDB0               LDIR
                 ;
B926 2159B9             LD   HL,FCHR4
B929 1122E7             LD   DE,FK4
B92C 010F00             LD   BC,000FH
B92F EDB0               LDIR
                 ;
B931 2168B9             LD   HL,FCHR5
B934 1132E7             LD   DE,FK5
B937 010600             LD   BC,0006H
B93A EDB0               LDIR
                 ;
B93C CD793F             CALL FCSUB
                 ;
B93F FF                 RST  38H
                 ;
B940 44423930 FCHR1: DB   'DB900.BFFF'
B944 302C4246
B948 4646
B94A 4C423930 FCHR2: DB   'LB900.BFFF'
B94E 302C4246
B952 4646
B954 41423930 FCHR3: DB   'AB900'
B958 30
```

```
B959 4D423930 FCHR4: DB    'MB900,BFFF,D900'
B95D 302C4246
B961 46462C44
B965 393030
B968 47423930 FCHR5: DB    'GB900',0DH
B96C 300D
                   ;
B96E                    END
```

●F220H番地からのダンプリスト

```
F220 3E FF 32 B8 E6 21 60 F2 11 F2 E6 01 0A 00 ED B0  > 2ク◥!`年 年◥    ○-
F230 21 6A F2 11 02 E7 01 0A 00 ED B0 21 74 F2 11 12  !j年  ◤    ○-!t年
F240 E7 01 05 00 ED B0 21 79 F2 11 22 E7 01 0F 00 ED  ◤    ○-!y年 "◤   ○
F250 B0 21 88 F2 11 32 E7 01 06 00 ED B0 CD 79 3F FF  -!! 年 2◤    ○-へy?
F260 44 42 39 30 30 2C 42 46 46 46 4C 42 39 30 30 2C  DB900,BFFFLB900,
F270 42 46 46 46 41 42 39 30 30 4D 42 39 30 30 2C 42  BFFFAB900MB900,B
F280 46 46 46 2C 44 39 30 30 47 42 39 30 30 0D FF 00  FFF,D900GB900
```

しかし，このプログラムでは，他のプログラムを走らせることができません．他のプログラムを走らせるためにはこのままのアドレスでは具合が悪いので，Sコマンドを使用して，F220H番地から16進数で入力します．その後で

"GF220 ⏎"

と入力しますとモニタに戻ります．$N_{88}$-BASICのワークエリアを壊していないので，どのような命令でも使用できます．

*à la carte 13*

# VRAMを移動させる

SRのV1モードでは，F3C8H番地から始まっているVRAMを動かせる．

N-BASICではテキストVRAMはF300H番地からと決まっていましたが，$N_{88}$-BASICでは自由に設定ができます．PC-8801シリーズでは，初期設定がF3C8H番地からになっているにすぎません．したがって，$N_{88}$-BASICのワークエリアを壊さない範囲ならVRAMを自由に設定できますが，コールド・リセットなどがかかるともとのF3C8H番地に戻ってしまいます．

VRAMのスタート・アドレスを決めているのは次の番地です．

E6C4H →下位アドレス
　(初期設定　C8H)
E6C5H →上位アドレス
　(初期設定　F3H)

## CRTコントローラを初期設定すればよい

このアドレスをSコマンドなどで適当に書き替えると，入力した文字は表示されません．これはCRTコントローラ（DMAを含む）が書き替えたアドレスを正しいアドレスとして参照しているからです．すなわち，CRTコントローラが初期設定されていないので表示できないのです．CRTコントローラを初期設定してやると，完全にVRAMエリアは変化します．初期設定の一部の画面モードのときに使用したものと全く同じものです．ただし，初期設定してしまうと，今までの文字はすべて消えます．

## VRAMスタート番地をD000H番地からに

VRAMのエリアを変化させて，スタート・アドレスをD000H番地からにするプログラムを作ってみましょう．手順は次のようにします．

①画面をクリアする
②E6C4HとF6C5H番地に新しいスタート・アドレスをセットする
③初期設定のルーチンを呼ぶ

これでOKです．プログラムを示します．

●テキストVRAMの移動

```
                 ;*************** COLOR TEXT MODE ***************
                 ;
                 ;TEXT VRAM ノ イトﾞウ (F3C8H-->D000H)
                 ;
                 ;PROGRAM NAME ---> MVRAM.e
                 ;
                 ;**********************************************
                 ;
                 ;
B900             STRAT: EQU  0B900H
5F0E             CLS:   EQU  5F0EH
E626             MON:   EQU  0E626H
6F6A             COLOR: EQU  6F6AH
                 ;
                        ORG  STRAT
                 ;
B900 CD0E5F             CALL CLS                ; CRT CLEAR
                 ;
B903 3E00               LD   A.00H              ;VRAM ADDRESS -->D000H
B905 32C4E6             LD   (0E6C4H),A
B908 3ED0               LD   A,0D0H
B90A 32C5E6             LD   (0E6C5H),A
                 ;
B90D 011950             LD   BC,5019H
B910 3E01               LD   A,01H
B912 32B2E6             LD   (0E6B2H),A
B915 3E19               LD   A,19H
B917 32B3E6             LD   (0E6B3H),A
B91A 3EE8               LD   A,0E8H
B91C 32B4E6             LD   (0E6B4H),A
B91F 3E00               LD   A.00H
B921 32B8E6             LD   (0E6B8H),A
B924 3EFF               LD   A,0FFH
B926 32B9E6             LD   (0E6B9H),A
B929 CD6A6F             CALL COLOR
                 ;
                 ;
```

```
B92C C326E6          JP    MON
           ;
B92F                 END
```

このプログラムを実行させると，VRAMのスタート・アドレスがD000H番地になります．したがって，F3C8H番地以降にキャラクタ・コードをセットしても，CRTには表示されません．Sコマンドを利用してD000H番地などにキャラクタ・コードを書き込んでみてください．コードに対応したキャラクタが表示されます．もちろん，マシン語ばかりでなくBASICモードでも，VRAMのスタート・アドレスはD000H番地となります．ためしにBASICモードにしてから，

Poke &HD000, &H41 ↵

としてください．左上に「A」が表示されます．

## 実験後は必ずリセットをかける

実験が終ったならば，必ずリセットをかけてください．リセットしないで，他のプログラムを実行させると暴走したり，正しく表示されません．必ず正しいVRAMのスタート・アドレスにセットする必要があるわけです．

VRAMのスタート・アドレスを自由に設定できるということは，テキスト画面を2画面持つことも可能なわけです．CRTコントローラ（DMAも含む）を初期設定しないようにすればできます．

*à la carte 14*

# ユーザが使える未使用命令

N88-BASICには，どんな未使用命令があり，どのように使えるか．

## 未使用命令は8個

N88-BASICでは，使用されていない未使用命令が8個あります．それは次のキーワードです．

**使用されていないキーワード**

① WBYTE
② RBYTE
③ POLL
④ ISET
⑤ IRESET
⑥ STATUS
⑦ IEEE
⑧ CMD

ただし，⑧のCMDは拡張命令の第1キーワードとして使用されているので使用できません．拡張命令を切り放してお

けば使用できます．

もともと，これらのキーワードはIEEEインターフェースの制御用としで用意されたものですが，$N_{88}$-BASICではサポートしていないので，未使用命令となっているのです．これらの命令を実行しても「Feature not available」というエラーメーッセージが表示されるだけで使用できません．しかし，これらのキーワードはジャンプテーブルを変えればユーザが自由に使用できます．これは，非常に便利な使い方です．たとえばPオプションの解除にはこの未使用命令を使用しないとできません．

## BASIC命令を使って未使用命令を実行

実際に独自の命令を作るとなると，結構時間を食われるので，BASIC命令を実行するようにしてみます．たとえば未使用命令の「POLL」をBASIC命令の「PRINT」命令にするということです．したがって，POLLを実行すればPRINT命令と全く同じに使用できます．これをうまく利用するとプログラムにプロテクトがかけられます．

未使用命令を使用するためには，それらのキーワードのジャンプテーブル番地を知る必要があります．それらは次のようになっています．

**キーワードのジャンプテーブル番地**

| キーワード | テーブル番地 |
|---|---|
| ① WBYTE | EEA1H |
| ② RBYTE | EEA4H |
| ③ POLL | EEA7H |
| ④ ISET | EEAAH |
| ⑤ IRESET | EEADH |
| ⑥ STATUS | EEB0H |
| ⑦ STATUS | EEB3H |
| ⑧ CMD | EEB6H |
| ⑨ IEEE | EEB9H |

⑥，⑦のSTATUSというキーワードは同じものですが，ジャンプアドレスが2つ用意されています．モニタを使用して，逆アセンブルしてみると次のようになります．

**●ジャンプアドレスを逆アセンブル**

```
                                  キーワード
---------------------------------------------
EEA1  C3 4DC1       JMP   4DC1  WBYTE
EEA4  C3 4DC1       JMP   4DC1  RBYTE
EEA7  C3 4DC1       JMP   4DC1  POLL
EEAA  C3 4DC1       JMP   4DC1  ISET
EEAD  C3 4DC1       JMP   4DC1  IRESET
EEB0  C3 4DC1       JMP   4DC1  STATUS
EEB3  C3 4DC1       JMP   4DC1  STATUS
EEB6  C3 4DC1       JMP   4DC1  CMD
EEB9  C3 4DC1       JMP   4DC1  IEEE
```

これらのジャンプテーブルにはすべて「JP 4DC1H」が入っています．ですからこれらのキーワードを実行するとすべて，4DC1H番地にジャンプします．4DC1H番地は「Feature not available」というエラーメッセージを表示するルーチンです．したがって4DC1H番地以外へジャンプすれば，これらの命令も使用できるようになります．

未使用命令を次のようなBASIC命令と対応させてみましょう．

| 未使用命令のキーワード | BASICのキーワード | BASICのエントリポイント |
|---|---|---|
| WBYTE | CLS | 71B5H |
| RBYTE | INPUT | 102DH |
| POLL | PRINT | 0E54H |
| ISET | RET | 0C9CH |
| IRESET | NEW | 77DDH |
| STATUS | LIST | 18D9H |
| CMD | RUN | 0B7CH |

●モニタを使って逆アセンブルしたもの

```
EEA1  C3 71B5    JMP  71B5
EEA4  C3 102D    JMP  102D
EEA7  C3 0E54    JMP  0E54
EEAA  C3 0C9C    JMP  0C9C
EEAD  C3 77DD    JMP  77DD
EEB0  C3 18D9    JMP  18D9
EEB3  C3 18D9    JMP  18D9
EEB6  C3 0B7C    JMP  0B7C
EEB9  C3 4DC1    JMP  4DC1
```

新しいジャンプ先が書かれている

●新しいキーワードで作ったプログラム

```
10 REM *** NEW KEY WORD TEST ***
20 WBYTE
30 RBYTE D
40 A=D*D
50 POLL "D*D= ";A
60 GOTO 30
70 END
```

●BASICコマンドで書き直すと…

```
10 REM *** NEW KEY WORD TEST (BASIC) ***
20 CLS
30 INPUT D
40 A=D*D
50 PRINT "D*D= ";A
60 GOTO 30
70 END
```

BASICのエントリポイントを未使用命令のジャンプテーブルにSコマンドなど使用して書き込みます．書き終ったならば，BASICモードに戻って，新しいキーワードでプログラムを作って実行させてみてください．しっかりと動くはずです．

下のプログラムを見てください．これは，新しいキーワードで作ったプログラムです．BASICに熟知している人であれば見当がつくかも知れませんが，ちょっと見ただけでは何のプログラムかわかりません．

これをBASICコマンドで書き直すと上のようになります．

未使用命令のジャンプテーブル群はリセット時に必ず4DC1Hが書き込まれますので，書き込んだデータはディスクに保存しておくと良いでしょう．

このジャンプテーブルをうまく利用すると，プログラムの実行キーワードを作って，他の人にプログラムを実行させないようにすることも可能です．自分で工夫してみてください．

（ヒント）

プログラムをロードした後にジャンプテーブルを書き替えるプログラムを実行させるか，BASICプログラム上に，あるキーワードでしか実行できないようなプログラムエリアを作っておくと良いでしょう．

*à la carte 15*

# プロテクトはずし

Pオプション付，すなわちプロテクトのかかったプログラムのリストをとる．

## 1 モニタにさえ入れない

Pオプション付のプログラムは，プログラムリストはもちろんモニタにさえも入ることはできません．Pオプションをつけると次のコマンドが実行できなくなります．

① LIST
② LLIST
③ MON
④ POKE
⑤ PEEK
⑥ BSAVE
⑦ BLOAD

Pオプションをつけるとリスト命令以外にマシン語を扱う命令までも実行できません．マシン語が実行できれば，簡単にPオプションを解除できます．

### 解除の原理は簡単

Pオプションの解除の原理は簡単で，次に示す番地へ00Hを書き込むだけで良いのです．

**ワークエリア**

EC29H

00H——→Pオプションなし

00H以外→Pオプションあり

（一般に20H）

したがって，EC29H番地に00Hを書き込めばよいのですが，前に説明したように，MON, POKE命令が使用できないので，書き込む方法がありません．Pオプションを解除するにはどうしてもマシン語のプログラムを実行させなければなりません．一番良いのは，モニタモードになってくれることなのですが……．

### キーワードは未使用命令

モニタモードに入る方法を考えましょう．もし，Pオプションがかかっていなければ，「MON⏎」で入ることができます．しかし，MON命令を使用せずにモニタに入る方法を考えなければなりません．いわゆる予約語以外の命令を実行させようとしても，「Syntax error」がでるだけです．したがって，BASICの予約語の中で，使用していない命令（未使用命令）を使用してみようというわけです．

未使用命令には次のような予約語（キーワード）があります．

**未使用の予約語（キーワード）とジャンプテーブル番地**

| | |
|---|---|
| ① WBYTE | EEA1H |
| ② RBYTE | EEA4H |
| ③ POLL | EEA7H |
| ④ ISET | EEAAH |
| ⑤ IRESET | EEADH |

⑥ STATUS　　EEB0H
⑦ STATUS　　EEB3H
⑧ CMD　　EEB6H
⑨ IEEE　　EEB9H

この中で，実用的に使用できるのは①～⑥までです．これらのジャンプテーブル番地にはすべて「JP 4DC1H」と書かれており，この4DC1H番地へジャンプしてしまいます．しかし，これらのジャンプテーブルは，RAM領域にあり，書き替えが可能です．そこで，このジャンプテーブルの飛び先番地をマシン語のプログラムの先頭にしてあげれば，モニタモードにすることが可能です．そこで，この未使用命令のうち2つを使用してみましょう．

POLL——モニタモードに入る
ISET——Pオプションを解除してBASICモードに戻る

これら2つの未使用命令の番地を書き替えて実行可能な命令にしますので，これらの命令を実行したときにエラーがでません．なお，未使用命令はマシン語のプログラムに入るキーとして使用しているだけです．

## 2 Pオプション解除プログラムを作る

ISET命令を使用してPオプションを解除するプログラムを作りましょう．

BASICではISET命令を実行すると，EEAAH番地に飛んできますので，ここの番地を次のように書き替えます．

EEAAH　JP　4DC1H
↓
JP　0F22AH

0F22AH番地からにEC29H番地を00Hにするプログラムを入れるだけです．プログラムとダンプリストを示します．

**●Pオプション解除**

```
                ;************** P OPTION FREE PROGRAM ***********
                ;
                ;P ｵﾌﾟｼｮﾝ ｦ ﾊｽﾞｽ
                ;
                ;PROGRAM NAME ---> POF.e
                ;
                ;**********************************************
                ;
F220            START: EQU  0F220H
50E5            REBAS: EQU  50E5H  ←— BASICへ戻る
                ;
                       ORG  START
                ;
F220 3E2A              LD   A,2AH         }
F222 32ABEE            LD   (0EEABH),A    } EEAAH→JP 0F22AH
F225 3EF2              LD   A,0F2H        } をセットする
F227 32ACEE            LD   (0EEACH),A    }
F22A 3E00              LD   A,00H
F22C 3229EC            LD   (0EC29H),A
F22F C3E550            JP   REBAS
                ;
F232                   END
```

**●ダンプリスト**

```
F220 3E 2A 32 AB EE 3E F2 32 AC EE 3E 00 32 29 EC C3
F230 E5 50 00
```

非常に短いプログラムですから，16進数で直接F220H番地から入力した方がよいでしょう．

## このプログラムを保存するには

プログラムの保存は次のようにしてください．BASICモードにしてから，

BSAVE "pof", &HF220, &H2FH ⏎

これで保存できます．このプログラムを実行する場合は，

BLOAD "pof", &HF220, r ⏎

とすると実行できます．

## プロテクトはずしの手順

このプログラムを使って，Pオプションを解除するには次のようにします．

① bload "pof", &HF220, r

で，pofプログラムを実行させてiset命令を生かす

② Pオプションのかかっているプログラム（この場合は「ptest」）をロード

③ iset命令を実行する

④ LISTをとることができる

このように非常に簡単に済むので，他のプログラムの解析に役立つでしょう．

●CRTの画面

```
bload "pof",&Hf220,r ―― ①
Ok
load "ptest" ―――――― ②
Ok
list
Illegal function call ) リストがとれない
Ok
iset ―― ③
Ok
list ―― ④
10 REM *********** P OPTION TEST *********
20 PRINT " **** P OPTION TEST PROGRAM *** "
30 PRINT
40 PRINT " P ｵﾌﾟｼｮﾝ ｦ ﾊｽﾞｽ ﾎｳﾎｳ ﾊ ｶﾝﾀﾝ ﾃﾞｽ "
50 PRINT
60 PRINT " ﾐﾅｻﾝ ﾓ ｺｺﾛﾐﾃ ｸﾀﾞｻｲ｡"
70 END
Ok
```

## Pオプションを解除せずにモニタに入る

このプログラムは先のISET命令のようにPオプションを解除しないで，モニタモードに入る方法です．モニタに入ったPオプションは解除されていませんから，BASICモードになればもちろんLISTはとることができません．

原理は簡単で，未使用命令のPOLL命令を再定義してF320H番地に飛ばし，そこでモニタモードに切り替えているだけなのです．

モニタモードにするためには，モニタROMがN-BASICモード上にあるために，I/Oポートの31Hを切り替えます．

**＊モニタに入る方法**

| | |
|---|---|
| PUSH AF | →Aレジスタを保存する |
| DI | →割り込みを禁止する |
| LD A, (0E6C2H) | →前にI/Oポート31Hに出力したデータをもってくる |
| OR 04H | →2ビット目を1にする（2ビット目が1のときにN-BASICモードとなる） |
| OUT (31H), A | →I/Oポート31Hに出力する |
| LD (0E6C2H), A | →I/Oポート31Hに出力したデータを次回のために保存する |
| EI | →割り込みを許可する |
| POP AF | →Aレジスタのデータを戻す |
| JP 6002H | →モニタROMのエントリポイントにジャンプする |

プログラムとダンプリストは次のように　なります.

**●モニタに入るプログラム**

```
                    ;*********** P OPTION TEST ON MONITOR ***********
                    ;
                    ; POLL COMMAND ON MONITOR
                    ;
                    ; PROGRAM NAME ---> POM.e
                    ;
                    ;**********************************************
                    ;
F320                START:  EQU   0F320H
E6C2                IOST:   EQU   0E6C2H
0031                IOAD:   EQU   31H
6002                MON:    EQU   6002H
                    ;
                            ORG   START
                    ;
F320 3E2B                   LD    A,2BH
F322 32A8EE                 LD    (0EEA8H),A
F325 3EF3                   LD    A,0F3H
F327 32A9EE                 LD    (0EEA9H),A
F32A C9                     RET
                    ;
F32B F5                     PUSH  AF
F32C F3                     DI
F32D 3AC2E6                 LD    A,(IOST)
F330 F604                   OR    04H
F332 D331                   OUT   (IOAD),A
F334 32C2E6                 LD    (IOST),A
F337 FB                     EI
F338 F1                     POP   AF
                    ;
F339 C30260                 JP    MON
                    ;
F33C                        END
```

**●ダンプリスト**

```
F320 3E 2B 32 A8 EE 3E F3 32 A9 EE C9 F5 F3 3A C2 E6
F330 F6 04 D3 31 32 C2 E6 FB F1 C3 02 60
```

このプログラムも長くありませんから，直接入力した方がよいでしょう．

プログラムの保存も先の「pof」と同じようにBASICモードにしてからBSAVEします．

BSAVE "pom", &HF320, &H2F ↵

## DUAD−88Dを使った場合

「pom」「pof」ともにN88-BASICのワークエリアの領域を使用しているために，DUADのLoaderを使用した場合には，他のディスケットにBSAVEできません．そこで，BSAVEする方法は，以下の通りになります．

①LOADで「pom」または「pof」をロード
②N88-BASICのシステムディスケットをドライブ1に入れて，リセットボタンを押す
③モニタモードのDコマンドを使用してプログラムが壊れていないか確認
④BASICモードに戻ってBSAVEする

## モニタに入ってPオプション解除

モニタモードに入ってしまえばPオプションを解除するのは楽なものです．

① "pom" をロードする
②Pオプションのついたプログラムをロードする

**●CRTの画面**

```
bload "pof",&Hf220,r——①
Ok
load "ptest"——②
Ok
list
Illegal function call
Ok
poll——③

h]sec29——④
EC29 20-00—⑤
h]^b——⑥
Ok

10 REM ************ P OPTION TEST **********
20 PRINT " **** P OPTION TEST PROGRAM *** "
30 PRINT
40 PRINT " P ｵﾌﾟｼｮﾝ ｦ ﾊｽﾞｽ ﾎｳﾎｳ ﾊ ｶﾝﾀﾝ ﾃﾞｽ "
50 PRINT
60 PRINT " ﾐﾅｻﾝ ﾓ ｺｺﾛﾐﾃ ｸﾀﾞｻｲ."
70 END
```

③POLL ↵ を実行する
④モニタモードのプロンプトがでるので，Pオプションフラグのある番地（EC29H）を調べる
⑤EC29H番地が20Hとなっているので，00Hを書き込む
⑥BASICモードに戻る．Pオプションがはずれているのでリストもとれる

いかがでしたか．Pオプションの解除のやり方はわかってしまうと非常に簡単ですね．BASICのコマンドだけでは絶対にPオプションは解除できません．

こんなところにもマシン語の威力があるのです．

*à la carte 16*

# 一気に移動させるブロック転送命令

Z-80CPUには任意のバイト数のデータを別のアドレスへ移動させる命令がある．

ブロック転送というのは，主に，メモリ上にあるデータを別の領域のメモリに移したいときに使用します．このブロック転送命令は数ステップのプログラムを1命令で実行し，スピードも速くよく利用されています．ブロック転送命令は4つ用意されており強力です．

ペアレジスタの用途は4つのブロック転送命令に共通で，次のようになります．

**●ブロック転送命令時の
ペアレジスタの用途**

HL ペアレジスタ
　――移動するメモリの番地
DE ペアレジスタ
　――データの行き先の番地
BC ペアレジスタ
　――バイト・カウンタ

**●使用されるフラグ**

P/V フラグ（Cy，S，Z フラグは変化しない）

## 1バイトごとのブロック転送 LDI命令

これは1バイトごとのブロック転送です．「JP，JR」などのジャンプ命令を用いてループを作ってブロック転送を行うものです．自分でループを作ることができますので，ループの間に何かの仕事をさせることができます．

ペアレジスタとフラグの動きは次のようになります．

**●ペアレジスタの動き**

HL＝HL＋1
DE＝DE＋1
BC＝BC－1

**●フラグの動き**

P/V フラグ
　実行した結果，BC ペアレジスタが0のとき0，1以上のとき1となる

HL ペアレジスタが指すメモリのデータをDE ペアレジスタが指すメモリに転送後 HL, DE ペアレジスタは1を加えられて，次の転送先の準備を行い，BC ペアレジスタから1を引きます．BC ペアレジスタに転送するバイト数を入れておけば自動的にカウントしてくれます．

また，P/V フラグは BC ペアレジスタが0になったときに0となるので，このフラグを調べれば，BC ペアレジスタが0になったかわかります．それには「JP PO，nn」，「JP PE，nn」などの命令を使用します．

VRAM の1行のデータを他の番地に転送するプログラムを作りましょう．

VRAM　　　――→　F3C8H
転送番地先　――→　D000H
転送バイト数――→　50H バイト

どうですか？　LDI命令を使用すると

●LDIを使用しなかったとき

```
      LD HL, 0F3C8H
      LD DE, 0D000H
      LD BC, 0050H
LOOP: LD A, (HL)
      LD (DE), A
      IND HL
      IND DE
      DEC BC
      LD A, B
      OR C
      JP NZ, LOOP
        ∫
```

●LDIを使用したとき

```
      LD HL, 0F3C8H
      LD DE, 0D000H
      LD BC, 0050H
LOOP: LDI
      JP PO, LOOP
        ∫
```

非常に短くなることがわかると思います．

このLDI命令は1バイトずつのブロック転送命令ですから，必ずジャンプ命令と組み合せて使用する必要があります．

## 連続してデータ転送するLDIR命令

このLDIR命令は連続してデータ転送をするものです．内部で自動的にループをしてくれるので，ループに必要なジャンプ命令などは必要ありません．したがって，フラグを調べる必要もありません．使い方はLDI命令と全く同じですが，ループが自動的なので，ループ間に別の仕事をはさむというわけにはいきません．

LDI命令を使用したサンプルプログラムとLDIRとを比較してみましょう．このように，ジャンプ命令がない分だけすっきりしたプログラムになります．LDI命令よりもこのLDIR命令を使用することが多いようです．

●LDI命令

```
      LD HL, 0F3C8H
      LD DE, 0D000H
      LD BC, 0050H
LOOP: LDI
      JP PO, LOOP
        ∫
```

●LDIR命令

```
LD HL, 0F3C8H
LD DE, 0D000H
LD BC, 0050H
LDIR
  ∫
```

## 大きい番地から小さい番地へデータ転送〜LDD命令

LDD命令はLDI命令と同じ1バイトごとのデータ転送命令なのですが，大き

く違うことはHL,DEペアレジスタともに1を引かれることです．LDI命令はアドレスの小さい方からデータ転送するのに対し，LDD命令はアドレスの大きい値から小さい値へとデータ転送することになります．

ペアレジスタとフラグの動きは次のようになります．

**●ペアレジスタの動き**

HL＝HL－1

DE＝DE－1

BC＝BC－1

**●フラグの動き**

P/V フラグ

実行した結果，BCペアレジスタが0のとき0，1以上のとき1

これらの動作については，HL，DEペアレジスタが－1になる他は同じです．

VRAMの1行のデータを他の番地に転送するプログラムを作ってみましょう．

VRAM ──→ F3C8H～F417H

転送番地 ──→ D000H～D04FH

転送バイト数──→50H

**●他の命令では**

```
        LD HL, 0F417H
        LD DE, 0D04FH
        LD BC, 0050H
LOOP：  LD A, (HL)
        LD  (DE), A
        DEC HL
        DEC DE
        DEC BC
        LD A, B
        OR C
        JP NZ, LOOP
          ∫
```

**● LDD命令を使用すると**

```
        LD HL, 0F417H
        LD DE, 0D04FH
        LD BC, 0050H
LOOP：  LDD
        JP PO, LOOP
          ∫
```

このようになります．LDI命令と違うところは，HL，DEペアレジスタにセットされるアドレスが，大きい方のアドレスで，小さい方のアドレスに向ってロードされていくことです．

## BCレジスタが0になるまで転送～LDDR命令

LDDR命令は連続してデータ転送をする命令です．内部で自動的にループを作ってくれますから，BCペアレジスタが0になるまで転送してくれます．

LDD命令で使用したサンプルプログラムと比較してみます．

この命令もジャンプ命令がない分だけプログラムがすっきりしています．

| ●LDD 命令 | ●LDDR 命令 |
|---|---|
| LD HL, 0F417H | LD HL, 0F417H |
| LD DE, 0D04FH | LD DE, 0D04FH |
| LD BC, 0050H | LD BC, LDDR |
| LOOP : LDD | LDDR |
| JP, LOOP | |

## 4つの命令を使い分ける

ブロック転送命令には「LDI, LDIR, LDD, LDDR」の4つの命令がありますが，実際には2つのグループに分けることができます.「LDI, LDIR」と「LDD, LDDR」の2つです.

それぞれの使い方について，メモリエリアを例にとって説明しましょう.

**①転送元のメモリエリアと転送先のメモリエリアが重ならない場合**

＊①4つの命令を自由に使えるとき

図①のように転送元のメモリエリアと転送先のメモリエリアが重ならない（オーバラップしない）ときは4つの命令を自由に使用できます.

**②転送元と転送先のメモリエリアが重なる（オーバラップする）場合**

オーバラップする場合ははっきりと2つのグループに違いがでてきます．つまり，どちらの方向に移動するかによって，どちらか一方しか使用できなくなります.

この違いをはっきりと認識していないと，転送元のデータを壊すことになります.

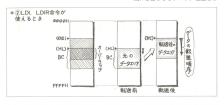
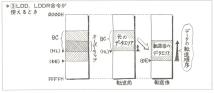
a LDI, LDIR 命令が使用できるとき.

図②のようにLDIとLDIR命令が使用できるのは必ず転送先のアドレスが転送元のアドレスよりも小さくなければならないことです．これは，転送先のアドレスが転送元のデータエリアのアドレスになっても，このときすでに，転送元のデータは新しいエリアに転送されているので，壊されても良いからです．LDI, LDIR 命令が使える条件は以下のとおりです.

LDI, LDIR が使える条件

HL ペアレジスタのデータ（転送元アドレス） ＞ DE ペアレジスタのデータ（転送先アドレス）

b LDD, LDDR 命令が使用できるとき

図③のように，現在のデータエリアをアドレスの大きい方に転送するには，アドレスの大きい方から転送しないと転送元のデータが壊れることが理解できると

思います．このように，アドレスの大きい方に転送できるようにしたのがLDD, LDDR命令です．LDD, LDDR命令が使える条件は次のようになります．

LDD，LDDR命令が使える条件

HLペアレジスタのデータ（転送元アドレス） ＜ DEペアレジスタのデータ（転送先アドレス）

ブロック転送命令を使用するときには，どちらの方向に転送するかを調べて，それに合った命令を使用してください．特にこれらの命令を有効に利用できるのはCRTへの書き込みやクリアのときで，高速化が図れます．

*à la carte 17*

# 微妙な中間色も出せるアナログモード

SRの特長のひとつの512色出せるアナログモードをマシン語で使うには.

## 512色出すのに必要なハードとソフト

SR の最大の特長の1つにアナログモードがあり，512色中から，任意の色を8色選択できるすごい機能です．せっかくマシン語を学んだわけですから，マシン語で操作してみたいと思います．ここでは，アナログモードで512色を出してみましょう．ただし，512色出すためには，次のハードウェアとモード設定が必要です．

**●ハードウエア**

アナログ RGB 用 CRT

**●ソフトウエア**

V 2モードでのみ可能．ただし，グラフィックスモードがモノクロの場合は，テキスト画面で使用できる

## 8レベルの明るさで色512色

デジタルモードでのカラーは一般的には8色までしか出せません．それは，CRT が赤，青，緑の色の組み合わせでできる色だけです．デジタル信号は常に0Vか5Vの2つに1つという原則にしたがっている関係上，2Vや3Vという中間の電圧は存在することが許されないからです．したがって，カラーCRT でも常に赤色が光っているか，光っていないかのどちらかしかないわけです．ですから3色の組み合わせでできる色は8色までということになります．RGB 専用 CRT はそれぞれの3色がよく出るように設計されていてるために，アナログモードのように使用したいと思っても無理です．

それに対してアナログモードでは発光源がデジタルモードと同じ3原色しかないのですが，それぞれの色の明るさを電圧の大きさで，コントロールすることによって，多くの色を出せます．SR のアナログモードでは各色の電圧を8レベルの大きさまでコントロールできるので，各色はそれに応じて8レベルの明るさをもつことになります．したがって，

赤　　　緑　　　青

8レベル×8レベル×8レベル＝512色

が出せるわけです．

それぞれの色のレベルは8レベルですから，2進数で表すと次のようになります．

8レベル＝$2^3$ビット

8レベルの明るさ（輝度という）をコントロールするためには，それぞれの色に3ビットですから，全部で9ビット必要になるわけです．

しかし，この9ビットは少しやっかいなビット数です．8ビットより1ビット

多いために2バイト必要となります．そこで，SRでは，mkIIなどとの互換性を考え，パレットI/Oポートに2回書き込むという方法をとっています．

少しやっかいなのですが，実に便利な使い方もあります．1つのパレットI/Oで512色出すことができるのです．このパレットI/OはデジタルモードとDI/Oアドレスを使用していますので，同時に出せるのは512色中8色までです．デジタルモードのカラーパレットは，3ビットデータのみ書き込んでいましたが，アナログモードのカラーパレットは，9ビットにデータを書き込めるので512色より選択が可能になったのです．

## カラーパレットの設定のし方

パレット番号とI/Oアドレスのデータは同じです．しかし，デジタルモードとアナログモードではカラーモードの設定が違うので注意が必要です．

●デジタルモード時のカラーパレットの設定

| I/O | パレット番号 | カラーデータ（初期設定時）<br>7 6 5 4 3 2 1 0ビット | 色 |
|---|---|---|---|
| 54H | 0 | —000———— | 黒 |
| 55H | 1 | —001———— | 青 |
| 56H | 2 | —010———— | 赤 |
| 57H | 3 | —011———— | 紫 |
| 58H | 4 | —100———— | 緑 |
| 59H | 5 | —101———— | 水色 |
| 5AH | 6 | —110———— | 黄色 |
| 5BH | 7 | —111———— | 白 |

このビットで決まる　—印は使用しない

●アナログモード時のカラーパレットの設定

| I/Oポート | パレット番号 | カラーデータ1<br>7 6 5 4 3 2 1 0ビット | カラーデータ2<br>7 6 5 4 3 2 1 0ビット | 色 |
|---|---|---|---|---|
| 54H | 0 | 01———000 | 00000000 | 黒 |
| 55H | 1 | 01———000 | 00000111 | 青 |
| 56H | 2 | 01———000 | 00111000 | 赤 |
| 57H | 3 | 01———000 | 00111111 | 紫 |
| 58H | 4 | 01———111 | 00000000 | 緑 |
| 59H | 5 | 01———111 | 00000111 | 水色 |
| 5AH | 6 | 01———111 | 00111000 | 黄色 |
| 5BH | 7 | 01———111 | 00111111 | 白 |

例1　00000001…一番暗い青（レベル1）
例2　00000111…一番明るい青（レベル7）

2．赤のレベルを設定するには

7 6 5 4 3 2 1 0ビット

| 0 | 0 | × | × | × | － | － | － |
|---|---|---|---|---|---|---|---|

操作しない
赤のレベルを設定するビット
必ず00にする

例1　00011000B…ちょっと明るい赤（レベル3）
例2　00111000B…一番明るい赤（レベル7）

3．緑のレベルを設定するには

7 6 5 4 3 2 1 0ビット

| 0 | 1 | － | － | － | × | × | × |
|---|---|---|---|---|---|---|---|

緑のレベルを
設定するビット
操作しない
必ず01にする

例1　01000010B…2番目に暗い緑（レベル2）
例2　01000100B…中間の明るさの緑（レベル4）

このように，512色から任意の色を出すためには，パレットのI/Oポートに必ず2回カラーデータを送らなければいけません．もし00……………Bにしかデータを送らないと，緑色を含まない色になってしまいます．

もし，パレット番号1に赤っぽい黄色を出すのであれば，

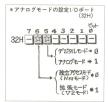
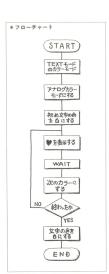
```
LD   A, 38H (00111000B)  ←┐
OUT(55H),A       赤を7レベルの輝度に
A, 44H (01000100B)  ←┐
OUT(55H),A       緑を4レベルの輝度に
```

というようになります．自分で好きな色が出せますが，CRT上の色と基準色とはずい分と違いがありますので，一度全部CRT上に表示してから決めるといいでしょう．

## テキストモードで512色出すプログラムを作る

実際にどのような色が出るのか，プログラムを作ってみてみましょう．このデモプログラムは色を見るためのものですからカラーに関係ない部分は簡単にします．このプログラムのフローチャートはP.287のようになっています．

色を変えていく方法はカラーデータの1とカラーデータの2から順番に1を引いています．

アナログモードにするにはI/Oポートの32Hにデータを書き込みます．

I/Oポート32Hの5ビット目と6ビット目はアナログモードが使用できるか，できないかを決めているスイッチです．他のビットはROMのバンク切り替えなどで使用していますので，勝手に変化させることはできません．このポート32HはIN命令が実行できますので，現在のモードを知ることができます．

マシン語で5ビット目と6ビット目だけ1にするには次のようにします．

```
IN A, (32H)  ←現在のデータを読み出す
OR 60H       ←60Hを加える．他のビットは変化しない
OUT(32H),A ← 5ビット目と6ビット目を1にしたデータを送る
```

もし，送ったデータが必要であれば，PUSH AF, POP AFで保存しておいてもよいでしょう．

なお，このテキスト画面のアナログカラーは，グラフィックス画面がモノクロ時のみ有効ですから，プログラムを実行する前に必ず次の命令を実行してください．

```
SCREEN 1, 0, 0, 7⏎
OK
```

なお，このプログラムは実行し終わるまでに約4分10秒かかります．プログラムはP.228にあります．

## もう少し色をゆっくり見たいとき

もう少し色をゆっくりと見たい場合は，WAITルーチンのデータを大きくしてください．B980Hの20Hを大きくすればよいのです．

また，このプログラムに多少手を加えて，カラーパレットに送っているデータを表示させると便利になります．D，Eレジスタがそれぞれの色のデータをもっているので，DISPのサブルーチンの中でDEレジスタのデータを表示させるようにします．数バイトで済みます．

## バックグランドも512色出せる

アナログモードでは，バックグランドの色も512色使用できます．色の出し方

●アナログモード時のバックグランドカラーの設定

| I/Oポート | 設定する色のレベル | カラーデータ<br>7 6 5 4 3 2 1 0ビット |
|---|---|---|
| 54H | BLUEのレベル | 1 0 ― ― ― × × × |
| 54H | REDのレベル | 1 0 × × × ― ― ― |
| 54H | GREENのレベル | 1 1 ― ― ― × × × |

ここは必ず1でなければならない

はP.285のテキストのアナログカラーと全く同じですが，カラーデータを設定するI/Oポートがデジタルモードとアナログモードとでは違います．デジタルモードでは52Hでしたが，アナログモードでは54Hになっています．この54Hはパレット機能で使用されています．7ビット目を1にすることによって，バックグランドのカラーデータかパレットのカラーデータかを区別しています．

ただし，他の55Hから5BHまでのI/Oポートはパレット機能のみですから，絶対に7ビット目を1にしないでください．もし，出力した場合には動作が保証されません．

デモプログラムはP.287と同じですから理解できると思います．

アナログモードの512色は比較的に美しいので，いろいろな，コンピュータグラフィックスなどにおおいに使用できます．グラフィックス時のパレットのデータ設定のやりかたは前と同じです．

●512色を順次表示するプログラム

```
                    ;*******************************************************
                    ;
                    ;PROGRAM MANE --> ANLGT  PC8801 MK2 SR
                    ;
                    ;TEXT ﾉ ｶﾗｰﾉ ﾃﾞﾓﾌﾟﾛｸﾞﾗﾑ  (ｱﾅﾛｸﾞ ﾓｰﾄﾞ)
                    ;
                    ;*******************************************************
                    ;
B900                START: EQU  0B900H
E626                MON:   EQU  0E626H
F3C8                ADCRT: EQU  0F3C8H
6F6B                COLOR: EQU  6F6BH               ;CRT COLOR SUBROUTINE
005B                PALT:  EQU  5BH                 ;ANLG PALET 7
3E9B                BELL:  EQU  3E9BH               ;BEEP SOUND
                    ;
                           ORG  START
                    ;
                    ;CRT MODE PROGRAM -------------------------
                    ;
B900 0650                  LD   B,50H           ;ﾖｺ ﾉ ｹﾀｽｳ ｦ ｾｯﾄ ﾄﾙ 0<Bﾚｼﾞｽﾀ>80
B902 0E19                  LD   C,19H           ;ﾀﾃ ﾉ ｷﾞｮｳ ｽｳ ｦ ｾｯﾄ ｽﾙ 0<Cﾚｼﾞｽﾀ>25
B904 3E01                  LD   A,01H           ;ｽｸﾛｰﾙ ｶｲｼ ｷﾞｮｳ
B906 32B2E6                LD   (0E6B2H),A
B909 3E19                  LD   A,19H           ;ｽｸﾛｰﾙ ｼｭｳﾘｮｳ ｷﾞｮｳ
B90B 32B3E6                LD   (0E6B3H),A
B90E 3EE8                  LD   A,0E8H          ;ｶﾗｰｺｰﾄﾞ
B910 32B4E6                LD   (0E6B4H),A
B913 3E00                  LD   A,00H           ;ﾌｧﾝｸｼｮﾝ ｺｰﾄﾞ ON,01H OFF.00H
```

```
B915 32B8E6            LD    (0E6B8H),A
B918 3EFF              LD    A,0FFH           ;ｶﾗｰ ﾏｰｸ ｶﾗｰ,FFH ﾓﾉｸﾛ,00H
B91A 32B9E6            LD    (0E6B9H),A
B91D CD6B6F            CALL  COLOR
                 ;
                 ;
                 ;
                 ; MAIN PROGRAM---------------------------
                 ;
B920 CD8B42      MAIN:  CALL  428BH           ;CURSOR OFF
                 ;
B923 DB32              IN    A,(32H)          ; ｱﾅﾛｸﾞ ﾓｰﾄﾞ ﾆｽﾙ
B925 F660              OR    60H
B927 D332              OUT   (32H),A
                 ;
B929 1647              LD    D,47H            ;GREEN LEVL
B92B 7A          LOOP2: LD    A,D
B92C D35B              OUT   (PALT),A
B92E 1E3F              LD    E,3FH            ; RED, BLUE LEVL
B930 7B          LOOP1: LD    A,E
B931 D35B              OUT   (PALT),A
B933 CD5DB9            CALL  DISP
B936 CD7DB9            CALL  WAIT
B939 1D                DEC   E
B93A C230B9            JP    NZ,LOOP1
B93D 7B                LD    A,E
B93E D35B              OUT   (PALT),A
B940 15                DEC   D
B941 7A                LD    A,D
B942 FE40              CP    40H
B944 D22BB9            JP    NC,LOOP2
                 ;
B947 CD9B3E            CALL  BELL
                 ;
B94A 3E3F              LD    A,3FH            ; ﾓﾆﾀｰ ﾆ ﾓﾄﾞﾙ ﾄｷ ﾉ ｶﾗｰ ｼﾛ
B94C D35B              OUT   (PALT),A
B94E 3E47              LD    A,47H
B950 D35B              OUT   (PALT),A
                 ;
B952 CD9042            CALL  4290H            ;CURSOL ON
                 ;
                 ;
B955 3E0C              LD    A,0CH            ;CRT CLEAR DATA 0CH
B957 CD1800            CALL  0018H            ;CRT CLEAR SUB
                 ;
                 ;
B95A C326E6            JP    MON
                 ;
                 ;
                 ;
                 ;
                 ;DISP SUB -----------------
                 ;
B95D E5          DISP:  PUSH  HL
B95E D5                PUSH  DE
B95F C5                PUSH  BC
B960 F5                PUSH  AF
B961 21C8F3            LD    HL,ADCRT
B964 112800            LD    DE,0028H
B967 0E19              LD    C,19H
B969 0650        DLOP2: LD    B,50H
B96B 3EE9              LD    A,0E9H           ;DISPLAY CHR ♥
B96D 77          DLOP1: LD    (HL),A
B96E 23                INC   HL
B96F 05                DEC   B
B970 C26DB9            JP    NZ,DLOP1
```

```
B973 19               ADD  HL,DE
B974 0D               DEC  C
B975 C269B9           JP   NZ,DLOP2
                 ;
                 ;
B978 F1               POP  AF
B979 C1               POP  BC
B97A D1               POP  DE
B97B E1               POP  HL
B97C C9               RET
                 ;
                 ;WAIT SUB -------------------
                 ;
B97D C5          WAIT: PUSH BC
B97E F5               PUSH AF
B97F 0E02             LD   C,02H
B981 06FF        WL3:  LD   B,0FFH
B983 3EFF        WL2:  LD   A,0FFH
B985 3D          WL1:  DEC  A
B986 C285B9           JP   NZ,WL1
B989 05               DEC  B
B98A C283B9           JP   NZ,WL2
B98D 0D               DEC  C
B98E C281B9           JP   NZ,WL3
B991 F1               POP  AF
B992 C1               POP  BC
B993 C9               RET
                 ;
B994                  END
```

●バックグランドカラーを512色出すプログラム

```
                 ;**************************************************
                 ;
                 ;PROGRAM MANE --> ANLGB  PC8801 MK2 SR
                 ;
                 ;TEXT ノ ｶﾗｰﾉ ﾃﾞﾓﾌﾟﾛｸﾞﾗﾑ (ﾊﾞｯｸｸﾞﾗﾝﾄﾞﾉｶﾗｰ) (ｱﾅﾛｸﾞ ﾓｰﾄﾞ)
                 ;
                 ;**************************************************
                 ;
B900             START: EQU  0B900H
E626             MON:   EQU  0E626H
F3C8             ADCRT: EQU  0F3C8H
6F6B             COLOR: EQU  6F6BH              ;CRT COLOR SUBROUTINE
0054             BAK:   EQU  54H                ;ANLG BACK GRAND COLOR
3E9B             BELL:  EQU  3E9BH              ;BEEP SOUND
                 ;
                        ORG  START
                 ;
                 ;CRT MODE PROGRAM -------------------------
                 ;
B900 0650              LD   B,50H       ;ﾖｺ ﾉ ｹﾀｽｳ ｦ ｾｯﾄ ﾄﾙ 0<Bﾚｼﾞｽﾀ>80
B902 0E19              LD   C,19H       ;ﾀﾃ ﾉ ｷﾞｮｳ ｽｳ ｦ ｾｯﾄ ｽﾙ 0<Cﾚｼﾞｽﾀ>25
B904 3E01              LD   A,01H       ;ｽｸﾛｰﾙ ｶｲｼ ｷﾞｮｳ
B906 32B2E6            LD   (0E6B2H),A
B909 3E19              LD   A,19H       ;ｽｸﾛｰﾙ ｼｭｳﾘｮｳ ｷﾞｮｳ
B90B 32B3E6            LD   (0E6B3H),A
B90E 3EE8              LD   A,0E8H      ;ｶﾗｰｺｰﾄﾞ
B910 32B4E6            LD   (0E6B4H),A
B913 3E00              LD   A,00H       ;ﾌｧﾝｸｮﾝ ｺｰﾄﾞ ON,01H OFF,00H
B915 32B8E6            LD   (0E6B8H),A
B918 3EFF              LD   A,0FFH      ;ｶﾗｰ ﾏｰｸ ｶﾗｰ,FFH ﾓﾉｸﾛ,00H
B91A 32B9E6            LD   (0E6B9H),A
```

```
B91D CD6B6F          CALL COLOR
             ;
             ;
             ;
             ; MAIN PROGRAM--------------------------
             ;
B920 CD8B42  MAIN:   CALL 428BH          ;CURSOR OFF
             ;
B923 DB32            IN   A,(32H)        ; ｱﾅﾛｸﾞ ﾓｰﾄﾞ ﾆｽﾙ
B925 F660            OR   60H
B927 D332            OUT  (32H),A
             ;
B929 16C7            LD   D,0C7H         ;GREEN LEVL
B92B 7A      LOOP2:  LD   A,D
B92C D354            OUT  (BAK),A
B92E 1EBF            LD   E,0BFH         ; RED, BLUE LEVL
B930 7B      LOOP1:  LD   A,E
B931 D354            OUT  (BAK),A
B933 CD55B9          CALL WAIT
B936 1D              DEC  E
B937 7B              LD   A,E
B938 FE80            CP   80H
B93A D230B9          JP   NC,LOOP1
B93D 7B              LD   A,E
B93E D354            OUT  (BAK),A
B940 15              DEC  D
B941 7A              LD   A,D
B942 FEC0            CP   0C0H
B944 D22BB9          JP   NC,LOOP2
             ;
B947 CD9B3E          CALL BELL
             ;
             ;
B94A CD9042          CALL 4290H          ;CURSOL ON
             ;
             ;
B94D 3E0C            LD   A,0CH          ;CRT CLEAR DATA 0CH
B94F CD1800          CALL 0018H          ;CRT CLEAR SUB
             ;
             ;
B952 C326E6          JP   MON
             ;
             ;WAIT SUB -------------------
             ;
B955 C5      WAIT:   PUSH BC
B956 F5              PUSH AF
B957 0E02            LD   C,02H
B959 06FF    WL3:    LD   B,0FFH
B95B 3EFF    WL2:    LD   A,0FFH
B95D 3D      WL1:    DEC  A
B95E C25DB9          JP   NZ,WL1
B961 05              DEC  B
B962 C25BB9          JP   NZ,WL2
B965 0D              DEC  C
B966 C259B9          JP   NZ,WL3
B969 F1              POP  AF
B96A C1              POP  BC
B96B C9              RET
             ;
B96C                 END
```

*à la carte 18*

# キーワードと中間言語を見る

N88-BASICのキーワードと中間言語を見るにはどうしたらよいか.

## BASICコマンドは中間言語に変換される

BASICインタプリタは，BASICのコマンドを必ず1バイトから2バイトのBASICインタプリタ専用の言語に変換してから実行しています．インタプリタ専用の言語を一般的に中間言語と呼んでいます.同じCPUであってもBASICインタプリタが違えば中間言語も違ってきます.

この中間言語は表には出てきませんが，BASICの構造を調べる上では役に立つ言語です.

また，この中間言語をうまく使用したコンパイラも発売されています．一般的に，Pコードコンパイラなどといいます.有名なのは,「UCSD PASCAL」などのP-SYSTEMです．Pコードコンパイラは完全なコンパイラではありませんので，インタプリタより速いという程度です.

キーワードと中間言語のコードをプリンタに印字する方法を考えましょう.

## 中間言語とキーワードの格納番地

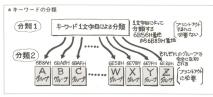
中間言語とキーワードを調べるには，ROM上のどこに，どのように格納されているかを知る必要があります.

中間言語とキーワードの分類は，キーワードの1文字をアルファベット順にしています.

この分類のやり方は2つの方法によって行われています．まず，キーワードの1文字目を調べて1文字目と同じアルファベットを使用しているグループに分けます．今度は同じグループ内で2文字目以降の文字を調べてキーワードを区別しています (P.293の図を参照のこと).

A～Zグループのアドレスは次のようになっています.

| | | | |
|---|---|---|---|
| A | 6B8AH | O | 6D4FH |
| B | 6BA0H | P | 6D68H |
| C | 6BAFH | Q | 6D97H |
| D | 6C0EH | R | 6D98H |
| E | 6C4AH | S | 6DDAH |
| F | 6C6FH | T | 6E21H |
| G | 6C89H | U | 6E41H |
| H | 6C9BH | V | 6E4AH |
| I | 6CA4H | W | 6E58H |
| J | 6CCEH | X | 6E7BH |
| K | 6CCFH | Y | 6E7FH |
| L | 6CDCH | Z | 6E80H |
| M | 6D1CH | | |
| N | 6D40H | | |

たとえばCを頭文字とするグループは6BAFH番地からあります.もし,"COPY"というキーワードの中間言語を知ろうと思ったならば,6BAFH～6C0DHの中にあ

るということです．

各グループと各グループの間には，00Hというデータが書かれています．00Hはよくセパレータとして使用されています．

## どのように格納されているか

キーワードと中間言語が格納されている番地はわかりましたが，これだけでは中間言語表をプリントアウトすることはできません．キーワードと中間言語がどのように格納されているかを知る必要があります．では，どのように格納されているかを調べてみましょう．

下のダンプリストは，6B8AH番地からをプリントアウトしたものの一部です．6B8AH番地は，Aを頭文字とするグループが入っている番地です．一番初めのキーワードは“AUTO”なのですが，これではよくわかりません．しかし，データ構造がわかると，すぐ区別がつきます．キーワードと中間言語の順番は，次のようになっています．

キーワード中間言語キーワード中間言語
（中間言語：1バイト）

これを見るとすぐわかるように，キーワードの後にすぐ中間言語がきています．次のキーワードとの間にも区切りを示す記号は何もありません．中間言語コードは1バイト構成と2バイト構成があります．しかし，このROMの中では2バイトのコードを強制的に1バイトで表しています．

では，1バイトコードと2バイトコードの格納のされ方を見てみましょう．

●6B8AH番地からのダンプリスト

```
6B8A 55 54 CF A8 4E C4 F8 42 D3 06 54 CE 0E 53 C3 15  UTﾏｨNﾄ Bﾓ Tﾎ Sﾃ
6B9A 54 54 52 A4 EB 00 53 41 56 C5 D5 4C 4F 41 C4 D4  TTR､♣ SAVﾅﾕLOAﾄﾔ
6BAA 45 45 D0 D7 00 4F 4E 53 4F 4C C5 9D 4F 50 D9 CD  EEﾐﾗ ONSOLﾅ OPﾙﾍ
6BBA 4C 4F 53 C5 C0 4F 4E D4 99 4C 45 41 D2 92 53 52  LOSﾅﾀONﾔ LEAﾒ SR
6BCA 4C 49 CE 54 49 4E D4 1C 53 4E C7 1D 44 42 CC 1E  LIﾎTINﾔ SNﾇ DBﾌ
6BDA 56 C9 20 56 D3 21 56 C4 22 4F D3 0C 48 52 A4 16  Vﾉ Vﾓ!Vﾄ"Oﾓ HR､
6BEA 41 4C CC B1 4F 4D 4D 4F CE B6 48 41 49 CE B7 4F  ALﾌｱOMMOﾎｶHAIﾎｷO
6BFA CD 5A 49 52 43 4C C5 CC 4F 4C 4F D2 CB 4C D3 CE  ﾍZIRCLﾅﾌOLOﾒﾋLﾓﾎ
```

**＊中間言語コード　1バイトの場合**

| | | | | | |
|---|---|---|---|---|---|
| キーワード | A | U | T | O | A8←中間言語コード |
| | ↓ | ↓ | ↓ | ↓ | ↓ |
| 理想的なデータ | 41 | 55 | 54 | 4F | A8 |
| | ↓ | ↓ | ↓ | ↓ | ↓ |
| 実際のデータ | | 55 | 54 | CF | A8 |
| | なし | | | | |

“A”というコードが入っていません．これは分類の1でキーワードの頭文字で分類しているので特に必要ないからです．また，“O”のアスキーコードが，4FH→CFHになっています．キーワードはすべてアルファベットからできているので，アスキーコードはすべて，7FH以下のデータになります．

そこでアルファベットのアスキーコードでは使用していない7ビット目（1×××××××B）を無理に使用しているのです．つまり7ビット目が1であるかないかによって，キーワードの終りの目印に使用しているのです．キーワードの長さはさまざまでどこでキーワードが終っているか区別がつかないので，キーワードの終りをこのビットが1になっているかどうかだけで知ることができるのです．本当ならば00Hなどのセパレータを1バイト入れておけばいいのですが，メモリを節約するために工夫しているのです．したがって，7ビット目が1のときは，次のデータが中間言語コードであることを表していることになります．

また，中間言語コードが1バイトなので，7ビット目が1になっているデータから2つ先のデータが，次のキーワードの2文字目から始まっていることがわかります．

中間言語コードは2バイトで表されているものもあります．中間言語コードが2バイト構成になっているものは，初めのデータが必ずFFHとなっており，意味があるのは2番目のデータだけです．では「ABS」命令を例にとってみましょう．

キーワードの表し方は，「AUTO」と全く同じです．しかし，中間言語コードは1バイトで表されています．よくみるとこのバイトで表しているデータも，加工されていることがわかります．2バイト

**＊中間言語コード　2バイトの場合**

| | | | | |
|---|---|---|---|---|
| キーワード | A | B | S | FF　86←中間言語コード |
| | ↓ | ↓ | ↓ | |
| 理想的なデータ | 41 | 42 | 53 | FF　86 |
| | | ↓ | ↓ | |
| 実際のデータ | | 42 | D3 | 06 |
| | なし | | | (86H－80＝06H) |

の中間言語コードを1バイトで表すために，7ビット目を強制的に0にしているのです．それによって，表される1バイトの中間言語は常に80Hよりも小さいということになります．したがって，80Hよりも小さい場合は2バイトの中間言語ということになります．このようにして，ここでもメモリの節約をしています．

## プリントアウトするために

キーワードとその中間言語をプリンタにプリントアウトするためには，次のようなことを知る必要があります．

a. プリンタにデータを出力する方法

b. 16進数を2桁のASCIIコードに直す方法

まず，この2つの方法について説明しましょう．

## 握手によく似たハンドシェーク法

CRT上にTEXT文字を表示させるには，VRAMにアスキーコードを書き込めばよかったのですが，プリンタにプリントアウトするには，プリンタのインターフェースにデータを送らなければなりません．送るタイミングをしっかり合わせてあげないと，データを送ってもプリンタは動いてくれません．データをプリンタに確実に転送する方法をハンドシェーク法と言います．ハンドシェーク法はデータ8ビット以外に2本の信号線をもっており，この2本の信号線でデータを確実に送っているのです．ハンドシェーク法の動きを簡単に説明しましょう．

まず，本体がプリンタにデータを送りたいと思ったら，データポート(I/O 10H)にデータをセットします．次に「今セットしたデータは正しいデータですよ」と，プリンタに信号を送ります(I/O OUT 40H 0ビット　ストローブ信号)．この信号(ストローブ信号)をプリンタが受けとり，本体からのデータを読みとります．プリンタは既に読みとっていたり，今読みとることができない場合は，本体に「今データを送られてもデータを読みとれないよ」という信号(I/O IN 40H 0ビット　ビジー信号)を送り返します．本体側は次のデータを送るときはビジー信号が出ていないことを確めてから，またデータを送っていきます．このように2本の信号をお互いに送り合って正しいデータの転送をしているのです．この2本の信号線は人間の握手によく似ていますので，このような名前がついています．

しかし，このような動きをするプログラム(ハンドラという)は自分で作るよりは，ROMルーチン内にありますので，そのルーチンを利用した方が便利です．

プリンタに1文字を送るROMルーチンは次のようになります．

＊プリンタに1文字送るROMルーチン

```
LD   A,  41H   ←プリンタに送るデータをAレジスタにセットする
CALL  0018H    ←プリンタに1文字を送るルーチンを呼ぶ
 (RST 18H)
```

注　プリンタスイッチがE64CHにあります．ここに00H以外のデータを書き込みます．もし，00HならばCRTに出力します．

プリンタに1文字を送って印字するサンプルプログラムは次のようになります。

```
LD   A, 20H     ←プリンタ出力モードにする
LD   (0E64CH), A
LD   A, 41H     ←Aコードをセットする
CALL  0018H
LD   A, 0AH     ←プリンタのヘッドをその行の初めに戻すコードを
CALL  0018H       プリンタに送る
LD   A, 0DH     ←改行させるコードをプリンタに送る
CALL  0018H
RST   38H       ←モニタに戻る
```

このプログラムを実行すると，“A”という文字を印字し改行します．0Aや0Dはコントロール・コードですが，このコードをプリンタに送らないと，プリンタは受けとったデータを印字しません．

## 16進数を2桁のアスキーコードに

キーワードはアスキーコードで格納されているので問題はありませんが，中間言語は16進数なのでそのままプリントアウトすることはできません．そこで，プログラムで16進数を2桁のアスキーコードにしなければなりません．

プログラムは下のようになります．このプログラムではBレジスタに上位の桁をセットしCレジスタに下位の桁をセットして戻ります．

このように，上位4ビットと下位4ビットを分けてアスキーコードを作っていきます．A8Hを例にとって説明するとP.297のようになります．

**●16進数を2桁のアスキーコードにするプログラム**

```
BA11 F5         ASCII:  PUSH AF ←下位4ビットのために保存
BA12 07                 RLCA  ⎫
BA13 07                 RLCA  ⎬ 右シフト命令．上位4ビットを
BA14 07                 RLCA  ⎪ 3ビット目～0ビット目にシフトする
BA15 07                 RLCA  ⎭
BA16 E60F               AND  0FH ←現在の上位4ビットを消す
BA18 CD24BA             CALL AS
BA1B 47                 LD   B,A ←Bレジスタにコピー
BA1C F1                 POP  AF  ←保存していたデータをとり出す
BA1D E60F               AND  0FH ←上位4ビットを消す
BA1F CD24BA             CALL AS
BA22 4F                 LD   C,A ←Cレジスタにコピー
BA23 C9                 RET
                ;
BA24 FE0A       AS:     CP   0AH  ⎫ 4ビットが9より大きいか調べる
BA26 DA2BBA             JP   C,AD ⎭
BA29 C607               ADS  A,07H ←9より大きければ07Hを加えて
BA2B C630       AD:     ADD  A,30H   A～Fのデータにする
BA2D C9                 RET  └── 下位4ビットに03Hを加えてアスキーコードにする
```

例　A8H

| | | 2進数で表すと |
|---|---|---|
| ①右へ4ビットシフトし，上位4ビットを消す | ——→0AH | 00001010B |
| ②9より大きいので07を加える | ——→0AH | 00001010B |
| | +)07H | +)00000111B |
| | 11H | 00010001B |
| ③11Hに30Hを加えると，41HというAのアスキーコードになった | 11H | 00010001B |
| | +)30H | +)00110000B |
| | 41H | 01000001B |
| ④もとのデータを読み出して，上位4ビットを消す | ——→08H | 00001000B |
| ⑤9より小さいのでそのまま30Hを加えると，38Hという8のアスキーコードになった | ——→08H | 00001000B |
| | +)30H | +)00110000B |
| | 38H | 00111000B |

## 2つの注意事項

このプログラムを作るときに注意しなければならないのは，キーワードの頭文字がすべてないことです．したがって，それぞれのグループに頭文字をつけてやる必要があります．

また印字するときに，印字用バッファを1キーワード分作っておき，次のキーワードに入る前に印字をします．中間言語のコードの印字位置を合わせるためにキーワードの次に00Hを入れておき，プリントアウトするルーチンがバッファ内の00Hを見つけると，プリンタのヘッドを10桁目へもってくるようにしています．バッファ内の終りの記号として00Hを使っており，00Hを見つけるともとのプログラムに戻ります．

なおプログラムについては，リストに説明してありますのでそちらをみてください．リストと実行結果を示します．

**●キーワードと中間言語を見るプログラム**

```
                    ;**************************************************
                    ;
                    ;KEYWORD LLIST
                    ;
                    ; PROGRAM NAME ---> KW.e
                    ;
                    ;**************************************************
                    ;
B900                START:  EQU   0B900H
0018                PNT:    EQU   0018H
6B8A                KWSAD:  EQU   6B8AH
                    ;
                    ;
                            ORG   START
                    ;
```

```
B900 ED7383BA        LD    (SPSAVE),SP
B904 31A3BA          LD    SP,STACK
                ;
B907 3E20            LD    A,20H            ;PRINTER ON
B909 324CE6          LD    (0E64CH),A
                ;
B90C CD58BA          CALL  CR
                ;
B90F DD2163BA        LD    IX,PSC
B913 218A6B          LD    HL,KWSAD
B916 1641            LD    D,'A'   ←―初めのキャラクタコードをセットする
B918 DD7200          LD    (IX),D
                ;
B91B 7A         LOOP: LD   A,D     ┐
B91C FE59            CP    59H     │ Yのキャラクタになったら，演算子を
B91E CAB6B9          JP    Z,OJP3  │ プリントアウトするプログラムへとばす
B921 7E              LD    A,(HL)  ┘
B922 FE00            CP    00H   ←―次のグループか調べる
B924 CAECB9          JP    Z,ZERO←―次のグループ処理プログラムへ
B927 FE80            CP    80H     ┐ 80Hより大きいときは，中間言語コードのセパレータ
B929 D235B9          JP    NC,OJP  ┘ も含んでいるのでそのプログラムへ行く
B92C DD23            INC   IX      ┐
B92E DD7700          LD    (IX),A  ┘ プリンタバッファに蓄える
B931 23              INC   HL   ←―次のアドレスにする
B932 C31BB9          JP    LOOP
                ;
B935 E67F       OJP:  AND  7FH←―セパレータを切り離す
B937 DD23            INC   IX
B939 DD7700          LD    (IX),A
B93C 3E00            LD    A,00H            ;SEPELETER 印字するために
B93E DD23            INC   IX                          入れておく
B940 DD7700          LD    (IX),A
B943 23              INC   HL      ┐
B944 7E              LD    A,(HL)  │ 中間言語が80Hより大きいか調べる．小さい
B945 FE80            CP    80H     │ 場合は2バイトコードの処理するプログラムへ
B947 DA75B9          JP    C,OJP2  ┘
B94A CDFFB9          CALL  ASCII←―中間言語をアスキーコードにする
B94D 78              LD    A,B
B94E DD23            INC   IX
B950 DD7700          LD    (IX),A
B953 79              LD    A,C
B954 DD23            INC   IX
B956 DD7700          LD    (IX),A
B959 3E0D            LD    A,0DH←―印字するときに終りを示すセパレータ
B95B DD23            INC   IX
B95D DD7700          LD    (IX),A
B960 CD1CBA          CALL  PRINT
B963 23              INC   HL
B964 7E              LD    A,(HL)
B965 FE00            CP    00H
B967 CA1BB9          JP    Z,LOOP
B96A DD2163BA        LD    IX,PSC←―バッファをクリアする
B96E 7A              LD    A,D     ┐
B96F DD7700          LD    (IX),A  ┘ 次の文字をバッファに送る
B972 C31BB9          JP    LOOP
                ;
B975 3E46       OJP2: LD   A,46H   ┐
B977 DD23            INC   IX      │
B979 DD7700          LD    (IX),A  │ 2バイトコードなので
B97C DD23            INC   IX      │ FFHをつける
B97E DD7700          LD    (IX),A  ┘
B981 3E20            LD    A,20H ←―スペースを空ける
B983 DD23            INC   IX
B985 DD7700          LD    (IX),A
B988 7E              LD    A,(HL)
B989 F680            OR    80H←―80Hを加えてもとの中間言語コードに戻す
```

```
B98B CDFFB9           CALL ASCII
B98E 78               LD   A,B
B98F DD23             INC  IX
B991 DD7700           LD   (IX),A
B994 79               LD   A,C
B995 DD23             INC  IX
B997 DD7700           LD   (IX),A
B99A 3E0D             LD   A,0DH
B99C DD23             INC  IX
B99E DD7700           LD   (IX),A
B9A1 CD1CBA           CALL PRINT
B9A4 23               INC  HL
B9A5 7E               LD   A,(HL)
B9A6 FE00             CP   00H
B9A8 CA1BB9           JP   Z,LOOP
B9AB DD2163BA         LD   IX,PSC
B9AF 7A               LD   A,D
B9B0 DD7700           LD   (IX),A
B9B3 C31BB9           JP   LOOP
                ;
B9B6 DD2163BA  OJP3:  LD   IX,PSC ←演算子をプリントアウトする
B9BA 7E               LD   A,(HL)
B9BB E67F             AND  7FH    ←セパレータを切り離す
B9BD DD7700           LD   (IX),A
B9C0 3E00             LD   A,00H               ;SEPELETER
B9C2 DD23             INC  IX
B9C4 DD7700           LD   (IX),A
B9C7 23               INC  HL
B9C8 7E               LD   A,(HL)
B9C9 CDFFB9           CALL ASCII
B9CC 78               LD   A,B
B9CD DD23             INC  IX
B9CF DD7700           LD   (IX),A
B9D2 79               LD   A,C
B9D3 DD23             INC  IX
B9D5 DD7700           LD   (IX),A
B9D8 3E0D             LD   A,0DH
B9DA DD23             INC  IX
B9DC DD7700           LD   (IX),A
B9DF CD1CBA           CALL PRINT
B9E2 23               INC  HL
B9E3 7E               LD   A,(HL)
B9E4 FE00             CP   00H
B9E6 CAF8B9           JP   Z,STOP ←全部印字したらストップ
B9E9 C31BB9           JP   LOOP
                ;
B9EC 14        ZERO:  INC  D      ←―――次の頭文字にする
B9ED DD2163BA         LD   IX,PSC
B9F1 DD7200           LD   (IX),D
B9F4 23               INC  HL
B9F5 C31BB9           JP   LOOP
                ;
                ;
B9F8 ED7B83BA  STOP:  LD   SP,(SPSAVE)
                ;
B9FC C33800           JP   0038H               ; GOTO MONITOR
                ;
B9FF F5        ASCII: PUSH AF                  ;ASCII CODE
BA00 07               RLCA
BA01 07               RLCA
BA02 07               RLCA
BA03 07               RLCA
BA04 E60F             AND  0FH
BA06 CD12BA           CALL AS
BA09 47               LD   B,A
BA0A F1               POP  AF
```

```
BA0B E60F              AND   0FH
BA0D CD12BA            CALL  AS
BA10 4F                LD    C,A
BA11 C9                RET
                 ;
BA12 FE0A        AS:   CP    0AH
BA14 DA19BA            JP    C,AD
BA17 C607              ADD   A,07H
BA19 C630        AD:   ADD   A,30H
BA1B C9                RET
                 ;
BA1C DD2163BA PRINT:   LD    IX,PSC              ;PRINTER OUT
BA20 1E00              LD    E,00H  ←―Eレジスタでプリンタヘッドの位置をカウント
BA22 DD7E00      PL2:  LD    A,(IX)
BA25 FE00              CP    00H
BA27 CA33BA            JP    Z,PL1
BA2A CD1800            CALL  PNT
BA2D DD23              INC   IX
BA2F 1C                INC   E
BA30 C322BA            JP    PL2
BA33 3E0A        PL1:  LD    A,0AH  ┐
BA35 93                SUB   E      │
BA36 47                LD    B,A    │ 中間言語コードが10桁目になると
BA37 3E20              LD    A,20H  │ 計算して20Hを出力する
BA39 CD1800      PL4:  CALL  PNT    │
BA3C 05                DEC   B      │
BA3D C239BA            JP    NZ,PL4 ┘
BA40 DD23        PLP:  INC   IX     ┐
BA42 DD7E00            LD    A,(IX) │ 0DH（終りの印）がくるまでバッファ
BA45 FE0D              CP    0DH    │ 内のデータを印字する
BA47 CA50BA            JP    Z,PL5  │
BA4A CD1800            CALL  PNT    ┘
BA4D C340BA            JP    PLP
BA50 CD58BA      PL5:  CALL  CR
BA53 DD2163BA          LD    IX,PSC
BA57 C9                RET
                 ;
BA58 3E0D        CR:   LD    A,0DH  ┐
BA5A CD1800            CALL  PNT    │
BA5D 3E0A              LD    A,0AH  │ 改行ルーチン
BA5F CD1800            CALL  PNT    │
BA62 C9                RET          ┘
                 ;
BA63             PSC:  DS    20H  ←―1行の文字を蓄えるバッファ
                 ;
BA83             SPSAVE:DS   02H
BA85                   DS    30
BAA3             STACK:
                 ;
BAA3                   END
```

## ●実行結果(中間言語コード)

| | | | | | |
|---|---|---|---|---|---|
| AUTO | A8 | CONSOLE | 9D | CVI | FF A0 |
| AND | F8 | COPY | CD | CVS | FF A1 |
| ABS | FF 86 | CLOSE | C0 | CVD | FF A2 |
| ATN | FF 8E | CONT | 99 | COS | FF 8C |
| ASC | FF 95 | CLEAR | 92 | CHR$ | FF 96 |
| ATTR$ | EB | CSRLIN | FF D4 | CALL | B1 |
| BSAVE | D5 | CINT | FF 9C | COMMON | B6 |
| BLOAD | D4 | CSNG | FF 9D | CHAIN | B7 |
| BEEP | D7 | CDBL | FF 9E | COM | FF DA |

| キーワード | 中間言語 |
|---|---|
| CIRCLE | CC |
| COLOR | CB |
| CLS | CE |
| CMD | FF E4 |
| DELETE | A7 |
| DATA | 84 |
| DIM | 86 |
| DEFSTR | AA |
| DEFINT | AB |
| DEFSNG | AC |
| DEFDBL | AD |
| DSKO$ | BA |
| DEF | 97 |
| DSKI$ | EC |
| DSKF | FF D0 |
| DATE$ | FF D9 |
| ELSE | 9F |
| END | 81 |
| ERASE | A3 |
| EDIT | A4 |
| ERROR | A5 |
| ERL | E4 |
| ERR | E5 |
| EXP | FF 8B |
| EOF | FF A3 |
| EQV | FB |
| FOR | 82 |
| FIELD | BC |
| FILES | C3 |
| FN | E1 |
| FRE | FF 8F |
| FIX | FF 9F |
| FPOS | FF A6 |
| GOTO | 89 |
| GO TO | 89 |
| GOSUB | 8D |
| GET | BD |
| HEX$ | FF 9A |
| HELP | D9 |
| INPUT | 85 |
| ISET | FF E0 |
| IEEE | FF E1 |
| IRESET | FF E2 |
| IF | 8B |
| INSTR | E8 |
| INT | FF 85 |
| INP | FF 90 |
| IMP | FC |
| INKEY$ | EF |
| KEY | FF DB |
| KILL | C5 |
| KANJI | DB |
| LOCATE | D6 |
| LPRINT | 9B |
| LLIST | 9C |
| LPOS | FF 9B |
| LET | 88 |
| LINE | AE |
| LOAD | C1 |
| LSET | C6 |
| LIST | 93 |
| LFILES | C9 |
| LOG | FF 8A |
| LOC | FF A4 |
| LEN | FF 92 |
| LEFT$ | FF 81 |
| LOF | FF A5 |
| MOTOR | FF D7 |
| MERGE | C2 |
| MOD | FD |
| MKI$ | FF A7 |
| MKS$ | FF A8 |
| MKD$ | FF A9 |
| MID$ | FF 83 |
| MON | CA |
| MAP | FF D5 |
| NEXT | 83 |
| NAME | C4 |
| NEW | 94 |
| NOT | E3 |
| OPEN | BB |
| OUT | 9A |
| ON | 95 |
| OR | F9 |
| OCT$ | FF 99 |
| OPTION | B8 |
| OFF | EE |
| PRINT | 91 |
| PUT | BE |
| POKE | 98 |
| POLL | FF DF |
| POS | FF 91 |
| PEEK | FF 97 |
| PSET | CF |
| PRESET | D0 |
| POINT | FF D3 |
| PAINT | D1 |
| PEN | FF D8 |
| RETURN | 8E |
| READ | 87 |
| RUN | 8A |
| RESTORE | 8C |
| RBYTE | FF DE |
| REM | 8F |
| RESUME | A6 |
| RSET | C7 |
| RIGHT$ | FF 82 |
| RND | FF 88 |
| RENUM | A9 |
| RANDOMIZE | B9 |
| ROLL | D8 |
| SCREEN | D3 |
| SEARCH | FF D6 |
| STOP | 90 |
| SWAP | A2 |
| SET | BF |
| SRQ | ED |
| STATUS | FF E3 |
| SAVE | C8 |
| SPC( | E2 |
| STEP | DF |
| SGN | FF 84 |
| SQR | FF 87 |
| SIN | FF 89 |
| STR$ | FF 93 |
| STRING$ | E6 |
| SPACE$ | FF 98 |
| THEN | DD |
| TRON | A0 |
| TROFF | A1 |
| TAB( | DE |
| TO | DC |
| TAN | FF 8D |
| TERM | D2 |
| TIME$ | FF DC |
| USING | E7 |
| USR | E0 |
| VAL | FF 94 |
| VIEW | FF D1 |
| VARPTR | EA |
| WIDTH | 9E |
| WINDOW | FF D2 |
| WAIT | 96 |
| WHILE | AF |
| WEND | B0 |
| WRITE | B5 |
| WBYTE | FF DD |
| XOR | FA |
|  | 00 |
| + | F3 |
| - | F4 |
| * | F5 |
| / | F6 |
| ^ | F7 |
| ¥ | FE |
| ' | E9 |
| > | F0 |
| = | F1 |
| < | F2 |

# 付録

# 80桁×25行モード VRAM番地表

| | 0 | | | | | 5 | | | | | 10 | | | | | 15 | | | | | 20 | | | | | 25 | | | | | 30 | | | | | 35 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | F3<br>C8 | C9 | CA | CB | CC | CD | CE | CF | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | DA | DB | DC | DD | DE | DF | E0 | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | EA | EB | EC | ED | EE | EF |
| | F4<br>40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 5A | 5B | 5C | 5D | 5E | 5F | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 |
| | F4<br>B8 | B9 | BA | BB | BC | BD | BE | BF | C0 | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | CA | CB | CC | CD | CE | CF | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | DA | DB | DC | DD | DE | DF |
| | F5<br>30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 3A | 3B | 3C | 3D | 3E | 3F | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 |
| | F5<br>A8 | A9 | AA | AB | AC | AD | AE | AF | B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 | B8 | B9 | BA | BB | BC | BD | BE | BF | C0 | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | CA | CB | CC | CD | CE | CF |
| 5 | F6<br>20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 2A | 2B | 2C | 2D | 2E | 2F | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 3A | 3B | 3C | 3D | 3E | 3F | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| | F6<br>98 | 99 | 9A | 9B | 9C | 9D | 9E | 9F | A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | AA | AB | AC | AD | AE | AF | B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 | B8 | B9 | BA | BB | BC | BD | BE | BF |
| | F7<br>10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D | 1E | 1F | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 2A | 2B | 2C | 2D | 2E | 2F | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 |
| | F7<br>88 | 89 | 8A | 8B | 8C | 8D | 8E | 8F | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 9A | 9B | 9C | 9D | 9E | 9F | A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | AA | AB | AC | AD | AE | AF |
| | F8<br>00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D | 1E | 1F | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 10 | F8<br>78 | 79 | 7A | 7B | 7C | 7D | 7E | 7F | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 8A | 8B | 8C | 8D | 8E | 8F | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 9A | 9B | 9C | 9D | 9E | 9F |
| | F8<br>F0 | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | FA | FB | FC | FD | FE | FF | F9<br>00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| | F9<br>68 | 69 | 6A | 6B | 6C | 6D | 6E | 6F | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 7A | 7B | 7C | 7D | 7E | 7F | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 8A | 8B | 8C | 8D | 8E | 8F |
| | F9<br>E0 | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | EA | EB | EC | ED | EE | EF | F0 | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | FA | FB | FC | FD | FE | FF | FA<br>00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 |
| | FA<br>58 | 59 | 5A | 5B | 5C | 5D | 5E | 5F | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 6A | 6B | 6C | 6D | 6E | 6F | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 7A | 7B | 7C | 7D | 7E | 7F |
| 15 | FA<br>D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | DA | DB | DC | DD | DE | DF | E0 | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | EA | EB | EC | ED | EE | EF | F0 | F1 | F2 | F3 | F4 | F5 | F6 | F7 |
| | FB<br>48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 5A | 5B | 5C | 5D | 5E | 5F | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 6A | 6B | 6C | 6D | 6E | 6F |
| | FB<br>C0 | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | CA | CB | CC | CD | CE | CF | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | DA | DB | DC | DD | DE | DF | E0 | E1 | E2 | E3 | E4 | E5 | E6 | E7 |
| | FC<br>38 | 39 | 3A | 3B | 3C | 3D | 3E | 3F | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 5A | 5B | 5C | 5D | 5E | 5F |
| | FC<br>B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 | B8 | B9 | BA | BB | BC | BD | BE | BF | C0 | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | CA | CB | CC | CD | CE | CF | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 |
| 20 | FD<br>28 | 29 | 2A | 2B | 2C | 2D | 2E | 2F | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 3A | 3B | 3C | 3D | 3E | 3F | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F |
| (行) | FD<br>A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | AA | AB | AC | AD | AE | AF | B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 | B8 | B9 | BA | BB | BC | BD | BE | BF | C0 | C1 | C2 | C3 | C4 | C5 | C6 | C7 |
| | FE<br>18 | 19 | 1A | 1B | 1C | 1D | 1E | 1F | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 2A | 2B | 2C | 2D | 2E | 2F | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 3A | 3B | 3C | 3D | 3E | 3F |
| | FE<br>90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 9A | 9B | 9C | 9D | 9E | 9F | A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | AA | AB | AC | AD | AE | AF | B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 |
| | FF<br>08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D | 1E | 1F | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 2A | 2B | 2C | 2D | 2E | 2F |

| 40 | | | | | 45 | | | | | 50 | | | | | 55 | | | | | 60 | | | | | 65 | | | | | 70 | | | | | 75（桁） | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F0 | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | FA | FB | FC | FD | FE | FF | F4<br>00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 68 | 69 | 6A | 6B | 6C | 6D | 6E | 6F | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 7A | 7B | 7C | 7D | 7E | 7F | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 8A | 8B | 8C | 8D | 8E | 8F |
| E0 | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | EA | EB | EC | ED | EE | EF | F0 | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | FA | FB | FC | FD | FE | FF | F5<br>00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 |
| 58 | 59 | 5A | 5B | 5C | 5D | 5E | 5F | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 6A | 6B | 6C | 6D | 6E | 6F | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 7A | 7B | 7C | 7D | 7E | 7F |
| D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | DA | DB | DC | DD | DE | DF | E0 | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | EA | EB | EC | ED | EE | EF | F0 | F1 | F2 | F3 | F4 | F5 | F6 | F7 |
| 48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 5A | 5B | 5C | 5D | 5E | 5F | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 6A | 6B | 6C | 6D | 6E | 6F |
| C0 | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | CA | CB | CC | CD | CE | CF | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | DA | DB | DC | DD | DE | DF | E0 | E1 | E2 | E3 | E4 | E5 | E6 | E7 |
| 38 | 39 | 3A | 3B | 3C | 3D | 3E | 3F | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 5A | 5B | 5C | 5D | 5E | 5F |
| B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 | B8 | B9 | BA | BB | BC | BD | BE | BF | C0 | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | CA | CB | CC | CD | CE | CF | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 |
| 28 | 29 | 2A | 2B | 2C | 2D | 2E | 2F | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 3A | 3B | 3C | 3D | 3E | 3F | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F |
| A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | AA | AB | AC | AD | AE | AF | B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 | B8 | B9 | BA | BB | BC | BD | BE | BF | C0 | C1 | C2 | C3 | C4 | C5 | C6 | C7 |
| 18 | 19 | 1A | 1B | 1C | 1D | 1E | 1F | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 2A | 2B | 2C | 2D | 2E | 2F | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 3A | 3B | 3C | 3D | 3E | 3F |
| 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 9A | 9B | 9C | 9D | 9E | 9F | A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | AA | AB | AC | AD | AE | AF | B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 |
| 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D | 1E | 1F | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 2A | 2B | 2C | 2D | 2E | 2F |
| 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 8A | 8B | 8C | 8D | 8E | 8F | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 9A | 9B | 9C | 9D | 9E | 9F | A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 |
| F8 | F9 | FA | FB | FC | FD | FE | FF | FB<br>00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D | 1E | 1F |
| 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 7A | 7B | 7C | 7D | 7E | 7F | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 8A | 8B | 8C | 8D | 8E | 8F | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 |
| E8 | E9 | EA | EB | EC | ED | EE | EF | F0 | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | FA | FB | FC | FD | FE | FF | FC<br>00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F |
| 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 6A | 6B | 6C | 6D | 6E | 6F | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 7A | 7B | 7C | 7D | 7E | 7F | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 |
| D8 | D9 | DA | DB | DC | DD | DE | DF | E0 | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | EA | EB | EC | ED | EE | EF | F0 | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | FA | FB | FC | FD | FE | FF |
| 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 5A | 5B | 5C | 5D | 5E | 5F | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 6A | 6B | 6C | 6D | 6E | 6F | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 |
| C8 | C9 | CA | CB | CC | CD | CE | CF | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | DA | DB | DC | DD | DE | DF | E0 | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | EA | EB | EC | ED | EE | EF |
| 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 5A | 5B | 5C | 5D | 5E | 5F | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 |
| B8 | B9 | BA | BB | BC | BD | BE | BF | C0 | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | CA | CB | CC | CD | CE | CF | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | DA | DB | DC | DD | DE | DF |
| 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 3A | 3B | 3C | 3D | 3E | 3F | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 |

# 40桁×25行モード VRAM番地表

| （行） | 0 | | | | | 5 | | | | | 10 | | | | | 15 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | F3<br>C8 | CA | CC | CE | D0 | D2 | D4 | D6 | D8 | DA | DC | DE | E0 | E2 | E4 | E6 | E8 | EA | EC | EE |
| | F4<br>40 | 42 | 44 | 46 | 48 | 4A | 4C | 4E | 50 | 52 | 54 | 56 | 58 | 5A | 5C | 5E | 60 | 62 | 64 | 66 |
| | F4<br>B8 | BA | BC | BE | C0 | C2 | C4 | C6 | C8 | CA | CC | CE | D0 | D2 | D4 | D6 | D8 | DA | DC | DE |
| | F5<br>30 | 32 | 34 | 36 | 38 | 3A | 3C | 3E | 40 | 42 | 44 | 46 | 48 | 4A | 4C | 4E | 50 | 52 | 54 | 56 |
| | F5<br>A8 | AA | AC | AE | B0 | B2 | B4 | B6 | B8 | BA | BC | BE | C0 | C2 | C4 | C6 | C8 | CA | CC | CE |
| 5 | F6<br>20 | 22 | 24 | 26 | 28 | 2A | 2C | 2E | 30 | 32 | 34 | 36 | 38 | 3A | 3C | 3E | 40 | 42 | 44 | 46 |
| | F6<br>98 | 9A | 9C | 9E | A0 | A2 | A4 | A6 | A8 | AA | AC | AE | B0 | B2 | B4 | B6 | B8 | BA | BC | BE |
| | F7<br>10 | 12 | 14 | 16 | 18 | 1A | 1C | 1E | 20 | 22 | 24 | 26 | 28 | 2A | 2C | 2E | 30 | 32 | 34 | 36 |
| | F7<br>88 | 8A | 8C | 8E | 90 | 92 | 94 | 96 | 98 | 9A | 9C | 9E | A0 | A2 | A4 | A6 | A8 | AA | AC | AE |
| | F8<br>00 | 02 | 04 | 06 | 08 | 0A | 0C | 0E | 10 | 12 | 14 | 16 | 18 | 1A | 1C | 1E | 20 | 22 | 24 | 26 |
| 10 | F8<br>78 | 7A | 7C | 7E | 80 | 82 | 84 | 86 | 88 | 8A | 8C | 8E | 90 | 92 | 94 | 96 | 98 | 9A | 9C | 9E |
| | F8<br>F0 | F2 | F4 | F6 | F8 | FA | FC | FE | F9<br>00 | 02 | 04 | 06 | 08 | 0A | 0C | 0E | 10 | 12 | 14 | 16 |
| | F9<br>68 | 6A | 6C | 6E | 70 | 72 | 74 | 76 | 78 | 7A | 7C | 7E | 80 | 82 | 84 | 86 | 88 | 8A | 8C | 8E |
| | F9<br>E0 | E2 | E4 | E6 | E8 | EA | EC | EE | F0 | F2 | F4 | F6 | F8 | FA | FC | FE | FA<br>00 | 02 | 04 | 06 |
| | FA<br>58 | 5A | 5C | 5E | 60 | 62 | 64 | 66 | 68 | 6A | 6C | 6E | 70 | 72 | 74 | 76 | 78 | 7A | 7C | 7E |
| 15 | FA<br>D0 | D2 | D4 | D6 | D8 | DA | DC | DE | E0 | E2 | E4 | E6 | E8 | EA | EC | EE | F0 | F2 | F4 | F6 |
| | FB<br>48 | 4A | 4C | 4E | 50 | 52 | 54 | 56 | 58 | 5A | 5C | 5E | 60 | 62 | 64 | 66 | 68 | 6A | 6C | 6E |
| | FB<br>C0 | C2 | C4 | C6 | C8 | CA | CC | CE | D0 | D2 | D4 | D6 | D8 | DA | DC | DE | E0 | E2 | E4 | E6 |
| | FC<br>38 | 3A | 3C | 3E | 40 | 42 | 44 | 46 | 48 | 4A | 4C | 4E | 50 | 52 | 54 | 56 | 58 | 5A | 5C | 5E |
| | FC<br>B0 | B2 | B4 | B6 | B8 | BA | BC | BE | C0 | C2 | C4 | C6 | C8 | CA | CC | CE | D0 | D2 | D4 | D6 |
| 20 | FD<br>28 | 2A | 2C | 2E | 30 | 32 | 34 | 36 | 38 | 3A | 3C | 3E | 40 | 42 | 44 | 46 | 48 | 4A | 4C | 4E |
| | FD<br>A0 | A2 | A4 | A6 | A8 | AA | AC | AE | B0 | B2 | B4 | B6 | B8 | BA | BC | BE | C0 | C2 | C4 | C6 |
| | FE<br>18 | 1A | 1C | 1E | 20 | 22 | 24 | 26 | 28 | 2A | 2C | 2E | 30 | 32 | 34 | 36 | 38 | 3A | 3C | 3E |
| | FE<br>90 | 92 | 94 | 96 | 98 | 9A | 9C | 9E | A0 | A2 | A4 | A6 | A8 | AA | AC | AE | B0 | B2 | B4 | B6 |
| | FF<br>08 | 0A | 0C | 0E | 10 | 12 | 14 | 16 | 18 | 1A | 1C | 1E | 20 | 22 | 24 | 26 | 28 | 2A | 2C | 2E |

| （行） | 20 | | | | | 25 | | | | | 30 | | | | | 35（桁） | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | F0 | F2 | F4 | F6 | F8 | FA | FC | FE | F4<br>00 | 02 | 04 | 06 | 08 | 0A | 0C | 0E | 10 | 12 | 14 | 16 |
| | 68 | 6A | 6C | 6E | 70 | 72 | 74 | 76 | 78 | 7A | 7C | 7E | 80 | 82 | 84 | 86 | 88 | 8A | 8C | 8E |
| | E0 | E2 | E4 | E6 | E8 | EA | EC | EE | F0 | F2 | F4 | F6 | F8 | FA | FC | FE | F5<br>00 | 02 | 04 | 06 |
| | 58 | 5A | 5C | 5E | 60 | 62 | 64 | 66 | 68 | 6A | 6C | 6E | 70 | 72 | 74 | 76 | 78 | 7A | 7C | 7E |
| | D0 | D2 | D4 | D6 | D8 | DA | DC | DE | E0 | E2 | E4 | E6 | E8 | EA | EC | EE | F0 | F2 | F4 | F6 |
| 5 | 48 | 4A | 4C | 4E | 50 | 52 | 54 | 56 | 58 | 5A | 5C | 5E | 60 | 62 | 64 | 66 | 68 | 6A | 6C | 6E |
| | C0 | C2 | C4 | C6 | C8 | CA | CC | CE | D0 | D2 | D4 | D6 | D8 | DA | DC | DE | E0 | E2 | E4 | E6 |
| | 38 | 3A | 3C | 3E | 40 | 42 | 44 | 46 | 48 | 4A | 4C | 4E | 50 | 52 | 54 | 56 | 58 | 5A | 5C | 5E |
| | B0 | B2 | B4 | B6 | B8 | BA | BC | BE | C0 | C2 | C4 | C6 | C8 | CA | CC | CE | D0 | D2 | D4 | D6 |
| | 28 | 2A | 2C | 2E | 30 | 32 | 34 | 36 | 38 | 3A | 3C | 3E | 40 | 42 | 44 | 46 | 48 | 4A | 4C | 4E |
| 10 | A0 | A2 | A4 | A6 | A8 | AA | AC | AE | B0 | B2 | B4 | B6 | B8 | BA | BC | BE | C0 | C2 | C4 | C6 |
| | 18 | 1A | 1C | 1E | 20 | 22 | 24 | 26 | 28 | 2A | 2C | 2E | 30 | 32 | 34 | 36 | 38 | 3A | 3C | 3E |
| | 90 | 92 | 94 | 96 | 98 | 9A | 9C | 9E | A0 | A2 | A4 | A6 | A8 | AA | AC | AE | B0 | B2 | B4 | B6 |
| | 08 | 0A | 0C | 0E | 10 | 12 | 14 | 16 | 18 | 1A | 1C | 1E | 20 | 22 | 24 | 26 | 28 | 2A | 2C | 2E |
| | 80 | 82 | 84 | 86 | 88 | 8A | 8C | 8E | 90 | 92 | 94 | 96 | 98 | 9A | 9C | 9E | A0 | A2 | A4 | A6 |
| 15 | F8 | FA | FC | FE | FB<br>00 | 02 | 04 | 06 | 08 | 0A | 0C | 0E | 10 | 12 | 14 | 16 | 18 | 1A | 1C | 1E |
| | 70 | 72 | 74 | 76 | 78 | 7A | 7C | 7E | 80 | 82 | 84 | 86 | 88 | 8A | 8C | 8E | 90 | 92 | 94 | 96 |
| | E8 | EA | EC | EE | F0 | F2 | F4 | F6 | F8 | FA | FC | FE | FC<br>00 | 02 | 04 | 06 | 08 | 0A | 0C | 0E |
| | 60 | 62 | 64 | 66 | 68 | 6A | 6C | 6E | 70 | 72 | 74 | 76 | 78 | 7A | 7C | 7E | 80 | 82 | 84 | 86 |
| | D8 | DA | DC | DE | E0 | E2 | E4 | E6 | E8 | EA | EC | EE | F0 | F2 | F4 | F6 | F8 | FA | FC | FE |
| 20 | 50 | 52 | 54 | 56 | 58 | 5A | 5C | 5E | 60 | 62 | 64 | 66 | 68 | 6A | 6C | 6E | 70 | 72 | 74 | 76 |
| | C8 | CA | CC | CE | D0 | D2 | D4 | D6 | D8 | DA | DC | DE | E0 | E2 | E4 | E6 | E8 | EA | EC | EE |
| | 40 | 42 | 44 | 46 | 48 | 4A | 4C | 4E | 50 | 52 | 54 | 56 | 58 | 5A | 5C | 5E | 60 | 62 | 64 | 66 |
| | B8 | BA | BC | BE | C0 | C2 | C4 | C6 | C8 | CA | CC | CE | D0 | D2 | D4 | D6 | D8 | DA | DC | DE |
| | 30 | 32 | 34 | 36 | 38 | 3A | 3C | 3E | 40 | 42 | 44 | 46 | 48 | 4A | 4C | 4E | 50 | 52 | 54 | 56 |

# ROM内システムサブルーチン

| 番地 | 項目名 | 機　能 |
| --- | --- | --- |
| 0000 | リセット | |
| 0008 | 文法をチェックする | RST（またはCALL）の後に置かれているキャラクタと(HL)と比較し，一致したらRST 10Hに戻り，一致しなかったら Syntax error を表示する |
| 0010 | テキストから1文字読み込む | テキストから1文字または数を読み込む |
| 0018 | 1文字出力 | CRTまたはLPTに1文字出力を行う．Aレジスタに文字コードを入れ RST 18H を実行する<br>(E64C) が 0 = CRT に出力<br>0 ≠ LPT に出力 |
| 0038 | ユーザ用 RST | モニタでは，ブレイクポイント用orコマンド待ちに戻るときに使う |
| 036F | コマンド待ちへ移る | スタックを実行前に戻してコマンド待ちへ移る |
| 047B | OKを出力する | OKを表示しコマンド待ちになる |
| 04A7 | コマンド待ち | コマンド待ちになる |
| 05BD | リンクポインタセット | テキストのリンクポインタをセットする．NEWしたプログラムを復活するのに必要である |
| 0B7C | RUN | RUNエントリ |
| 0C9C | LET | LET文エントリ |
| 18D9 | LIST | LISTエントリ |
| 3242 | キースキャン | キースキャンを行う<br>Cy＝1：キーは押されていない |

| 番地 | 項目名 | 機　能 |
|---|---|---|
| | | Cy＝0, Z＝1：キーを押したまま<br>Cy＝0, Z＝1：新しいキーを押した<br>（EEFEHにデータ） |
| 3583 | キー入力 | キーボード（キー入力用キュー）から1文字入力してAレジスタに入れ，入力待ち |
| 35C2 | ブレイクのチェック | STOP, ^C のときキャリーフラグをセットして戻る |
| 35CE | キー入力 | キーバッファから1文字読む<br>Z＝1：バッファエンプティ<br>Z＝0：Aレジスタに入力データが代入されている |
| 3E0D | CRT出力 | CRTへAレジスタの1文字を出力する<br>（Aレジスタ＝ASCIIコード） |
| 3E9B | BELL | BEEP出力 |
| 3EC0 | BEEP ON/OFF | Aレジスタ＝0：BEEP0<br>Aレジスタ≠0：BEEP1 |
| 3F79 | ファンクションキーの表示 | ファンクションキー（[f･1]～[f･5]）の表示をする<br>(E6B8) ＝0：表示しない<br>(E6B8) ≠0：表示する |
| 3F7A | ファンクションキーの表示 | ファンクションキーの表示<br>(E6B8) ＝0：表示しない<br>Aレジスタ＝0：[f･1]～[f･5]<br>Aレジスタ＝5：[f･6]～[f･10]<br>（を表示する） |
| 4021 | 最下行をクリア | ファンクションキーの表示を消し，テキスト画面の最下行をクリアする |
| 4047 | タイマからデータを読み込む | タイムバッファに時間のデータを読み込む<br>F00DH←秒（BCDコード）<br>F00EH←分（BCDコード） |

| 番地 | 項目名 | 機　能 |
|---|---|---|
| | | F00FH←時（BCDコード）<br>F010H←日（BCDコード）<br>F011H←月（バイナリコード）<br>F012H←年（BCDコード） |
| 428B | カーソルを消す | カーソルを表示しない |
| 4290 | カーソルを表示 | カーソルを表示する |
| 429D | VRAMのアドレスを求める | キャラクタ座標（H, L）が，どのアドレスに対応しているかを計算する<br>レジスタHLに与える座標の範囲が不適正なとき，“Illegal function call”が発生 |
| 4F01 | NEW | NEWを実行する |
| 4F21 | CLEAR | 変数のCLEARを実行する |
| 5550 | 文字列出力 | HLレジスタの示すアドレスからデータ00Hまでの文字列を出力する |
| 559A | ガベージコレクション | ガベージコレクションを実行する |
| 5935 | プリンタ出力 | プリンタへ1文字出力する |
| 5989 | プリンタ改行 | LSOP≠0（プリンタヘッドが行の先頭にないとき）プリンタを改行する |
| 69FE | 処理ルーチンのアドレステーブル | 中間言語81H～D9Hの処理ルーチンアドレステーブル |
| 6AB0 | 処理ルーチン（FFシリーズ前半）のアドレステーブル | 中間言語FF81～FFA9Hの関数の処理ルーチンアドレステーブル |
| 6B02 | 処理ルーチン（FFシリーズ後半）のアドレステーブル | 中間言語FFD0～FFE4Hの関数・文としての処理ルーチンアドレステーブル<br>1組4バイト<br>前半2バイト：関数の処理ルーチンアドレス |

| 番地 | 項目名 | 機　能 |
| --- | --- | --- |
| | | 後半2バイト：文の処理ルーチンアドレス |
| 6B56 | 予約語インデックス | 予約語テーブルの最初1文字による INDEX |
| 6B8A | 予約語テーブル | 予約語と中間言語のリストが入っている |
| 6E81 | 演算子テーブル | 1文字の予約語と中間言語のリスト |
| 6E96 | E3ROM 処理呼び出し | E3ROM 処理を呼び出すために使用 |
| | | 1組4バイト |
| | | CALL　4551H |
| | | DB　〈処理#〉 |
| | | これを実行すれば，〈処理#〉に対する処理が実行される |
| 6F6B | テキスト画面 WIDTH | テキスト画面のモード変更 |
| | | エントリ：B←桁数 |
| | | C←行数 |
| | | E6B2H　スクロール開始行 |
| | | E6B3H　スクロール終了行 |
| | | E6B4H　ヌルアトリビュート |
| | | E6B9H　{00H：白黒モード / FFH：カラーモード} |
| 6FD1 | CRTC セット | CRTC，DMAC をセット |
| | | エントリ： |
| | | HL レジスタ＝705BH(20行)，7066H(25行) |
| | | (E6C4，5H)＝VRAM 先頭アドレス(F3C8H) |
| | | (E6B9H)　＝00H(白黒)，FFH(カラー) |
| | | Cyフラグ　＝1(40桁)，0(80桁) |

# RAM上のシステムワークエリア

| 番地 | バイト数 | 機能・用途 |
|---|---|---|
| E64B | 1 | LPOSの値（プリンタヘッドの位置） |
| E64C | 1 | プリンタフラグ（出力先の指定．0=CRT, 0≠LPT） |
| E64D | 1 | LPRINT'，プリンタの横幅 |
| E64E | 1 | WIDTH LPRINTで指定した値(プリンタの横幅を示す値が入っている) |
| E652 | 1 | CTRL＋Oフラグ（画面出力をさせないためのもの） |
| E654 | 2 | フリーエリアの上限値，CLEAR文の第2パラメータ |
| E656 | 2 | 実行中の行番号 |
| E658 | 2 | テキストの開始アドレス(BASICプログラム開始番地) |
| E669 | | モニタ用ワークエリア |
| E69C | 2 | RST 10H用ワーク（1EH, 10H） |
| E6A8 | 1 | カーソル表示ON/OFFコマンド |
| E6B0 | 1 | テキスト画面ウィンドウ上限（スクロール開始行＋1） |
| E6B1 | 1 | テキスト画面ウィンドウ下限（スクロール終了行） |
| E6B4 | 1 | ヌルアトリビュートコード |
| E6B5 | 1 | ヌルキャラクタコード |
| E6B6 | 1 | CRTにコントロールコードを出力する |
| E6B7 | 1 | 未使用 |
| E6B8 | 1 | ファンクションキーを表示するフラグ |
| E6B9 | 1 | テキストモード（00H：白黒，FFH：カラー） |
| E6C0 | 1 | ポート30Hへの出力データ |
| E6C1 | 1 | ポート40Hへの出力データ |
| E6C2 | 1 | ポート31Hへの出力データ |
| E6C3 | 1 | ポートE4Hへの出力データ |
| E6C4 | 2 | テキストVRAM先頭アドレス（初期値：F3C8H） |
| E6F2 | 16 | ファンクションキー f･1 のデータ |

| 番地 | バイト数 | 機能・用途 |
|---|---|---|
| E702 | 16 | ファンクションキー f・2 のデータ |
| E712 | 16 | ファンクションキー f・3 のデータ |
| E722 | 16 | ファンクションキー f・4 のデータ |
| E732 | 16 | ファンクションキー f・5 のデータ |
| E742 | 16 | ファンクションキー f・6 のデータ |
| E752 | 16 | ファンクションキー f・7 のデータ |
| E762 | 16 | ファンクションキー f・8 のデータ |
| E772 | 16 | ファンクションキー f・9 のデータ |
| E782 | 16 | ファンクションキー f・10 のデータ |
| E826 | | モニタ用サブルーチン |
| EAC0 | 2 | RST 10H 用テキストポインタ |
| EAC2 | 1 | RST 10H で読んだ文字 |
| EAC3 | 1 | RST 10H 用 FAC のデータのタイプ |
| EAC4 | 8 | RST 10H 用 FAC のデータ |
| EC27 | 1 | P オプションプログラムをセーブ中であることを示す |
| EC28 | 1 | P オプションプログラムをロード中であることを示す |
| EC29 | 1 | P オプション実行フラグ(0以外でLIST POKE MON SAVE がエラーとなる) |
| ECBB | 1 | フックアドレステーブル |
| EF85 | 1 | 1行入力最後の桁 |
| EF86 | 1 | カーソルの Y 座標 |
| EF87 | 1 | カーソルの X 座標 |
| EF88 | 1 | テキスト画面の行数 |
| EF89 | 1 | テキスト画面の桁数 |
| EF90 | 1 | ファンクションキー表示開始番号（00H：ノーマル，05 H：シフト |
| EF9A | | テキスト画面行継続コード（00H：つながっている，0A H：^J，その他：つながっていない） |

| 番地 | バイト数 | 機能・用途 |
|---|---|---|
| F00D | 6 | 時間用バッファ<br>F00DH←秒（BCDコード）<br>F00EH←分（BCDコード）<br>F00FH←時（BCDコード）<br>F010H←日（BCDコード）<br>F011H←月（バイナリコード）<br>F012H←年（BCDコード） |
| F01E | 1 | フォアグラウンドカラー |
| F01F | 1 | バックグラウンドカラー |
| F044 | 1 | PAINT 文：領域色パレット番号，LINE文：色パレット番号 |
| F21E | 226 | 未使用 |
| F320 | 168 | 未使用 |
| FFF8 | 8 | 未使用 |

＊未使用の部分は，ユーザが勝手に使用できるためよく使用される．ここを利用すると，テキストエリアを使わずに済む．

# システムI/Oポート

| 番地 | I/O | 内容 |
|---|---|---|
| 30 | O | システムコントロールポート |

MSB　　　　　　　　　　　　　　　　　　LSB

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ＼ | ＼ | BS2 | BS1 | MTON | CDS | $\overline{\text{COLOR}}$ | $\overline{40}$ |

$\overline{40}$……………40桁モードか否かを設定．このビットを直接操作することによって画面の桁数が変えられるわけではないので注意すること．テキスト VRAM の内容の読み出し方を変える

0：40桁モード

1：80桁モード

$\overline{\text{COLOR}}$………カラーモードか否かを設定

0：カラーモード

1：白黒モード

CDS…………カセットへ出力する信号の切り替え

MTON………カセットのモータの ON(1), OFF(0)

BS1, BS2…USART をカセットインターフェースに使うか, RS-232C に使うかを選択

| 番地 | I/O | 内容 |
|---|---|---|
| 30 | I | モードセレクト |

MSB　　　　　　　　　　　　　　　　　　LSB

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ＼ | ＼ | SW1 S5 | SW1 S4 | SW1 S3 | SW1 S2 | SW1 S1 | SW4 S1 |

ディップスイッチの値を読み込む

0：ON

1：OFF

| 番地 | I/O | 内　容 |
|---|---|---|
| 31 | O | グラフィックコントロール |

MSB　　LSB

| | | 25 LINE | HCOLOR | GRPH | RMODE | MMODE | 200 LINE |
|---|---|---|---|---|---|---|---|

200LINE……専用高解像ディスプレイモードのコントロール
0：400Line
1：200Line

MMODE………RAMモードコントロール
0：ROM, RAMモード
1：ALL RAMモード（64K RAM）
このビットが0のときは一切のROMが禁止される．
このモードはCP/Mなどのアプリケーションを走らせる場合に使用する

RMODE………ROMモードコントロール
0：$N_{88}$-BASIC
1：N-BASIC
選択されるROMのバンク，テキストウィンドウの可不可などを選択する

GRPH…………グラフィックディスプレイモード
0：表示禁止
1：表示

HCOLOR……カラーグラフィックディスプレイモード
0：白黒
1：カラー
単色3ページモードか，カラー1ページモードかの選択をする

25LINE………専用高解像ディスプレイLINEコントロール
0：20LINEモード
1：25LINEモード

| 番地 | I/O | 内　容 |
|---|---|---|
| 31 | I | モードセレクト |

MSB　　　　　　　　　　　　　　　　　　　　　　　　LSB

| SW4 S2 | SW3 S0 | SW2 S6 | SW2 S5 | SW2 S4 | SW2 S3 | SW2 S2 | SW2 S1 |
|---|---|---|---|---|---|---|---|

ディップスイッチの内容を取り込む

0 ：ON

1 ：OFF

| 番地 | I/O | 内　容 |
|---|---|---|
| 32 | I/O | モード指定 |

MSB　　　　　　　　　　　　　　　　　　　　　　　　LSB

| SINTM | GVAM | PMODE | TMODE | AVC2 | AVC1 | EROM SL1 | EROM SL0 |
|---|---|---|---|---|---|---|---|

| EROMSL1 | EROMSL0 | ……………$N_{88}$-BASIC で拡張 ROM の番号を選ぶときに使用する |
|---|---|---|
| 0 | 0 | ：内部 EROM バンク 0 を選択 |
| 0 | 1 | ：内部 EROM バンク 1 を選択 |
| 1 | 0 | ：内部 EROM バンク 2 を選択 |
| 1 | 1 | ：内部 EROM バンク 3 を選択 |

| AVC2 | AVC1 | ……………AV コントロール |
|---|---|---|
| 0 | 0 | ：テレビ／ビデオモード |
| 0 | 1 | ：―――――――― |
| 1 | 0 | ：アナログ RGB モード |
| 1 | 1 | ：オプションモード |

TMODE …………高速 RAM 選択

0 ：高速 RAM

1 ：メイン RAM

高速モード時にはメインRAMのF000～FFFFH番地までが高速RAMにも切り替えられる．どちらの RAM を選択するか決めるビット

<table>
<tr><th>番地</th><th>I/O</th><th>内　容</th></tr>
<tr><td>32</td><td>I/O</td><td>PMODE…………カラーモード<br>0：ディジタルモード<br>1：アナログモード<br>カラーパレットの内容がディジタルRGBか，アナログRGBかを決める<br>GVAM……………G-VRAMアクセスモード<br>0：独立アクセスモード<br>1：拡張アクセスモード<br>PC-8801mkIISRで採用されたALUによる画面アクセスモードか，従来どおりの3回に分けてアクセスするモードかを決める（ALU：Arithmetic Logic Unit）<br>SINTM…………PSG割り込みマスク<br>0：割り込み許可<br>1：割り込み禁止<br>PSGを使ってバックグランドで音を出すか出さないかの制御</td></tr>
<tr><td>34</td><td>O</td><td>G-VRAM制御<br><br>MSB　　　　　　　　　　　　LSB<br><table><tr><td></td><td>ALU 21</td><td>ALU 11</td><td>ALU 01</td><td></td><td>ALU 20</td><td>ALU 10</td><td>ALU 00</td></tr></table>0：ALUn1，0：ALUn0……ビットリセット<br>0：ALUn1，1：ALUn0……ビットセット<br>1：ALUn1，0：ALUn0……ビット反転<br>1：ALUn1，1：ALUn0……ノーオペレーション<br>(n＝0～2)<br>●拡張G-VRAMアクセスモードで各ページのALUにどのような演算をさせるかを指定する。</td></tr>
</table>

| 番地 | I/O | 内　容 |
|---|---|---|
| 35 | O | G-VRAM制御 |

MSB　LSB

| GAM | ＼ | GDM1 | GDM0 | ＼ | PLN2 | PLN1 | PLN0 |
|---|---|---|---|---|---|---|---|

PLN0……………G-VRAM0の比較データ
PLN1……………G-VRAM1の比較データ
PLN2……………G-VRAM2の比較データ

| GDM1 | GDM0 | …グラフィックデータマルチプレクサ制御 |
|---|---|---|
| 0 | 0 | …ALUの出力を同時ライト |
| 0 | 1 | …リードデータを同時ライト |
| 1 | 0 | …G-VRAM1→G-VRAM0に転送 |
| 1 | 1 | …G-VRAM0→G-VRAM1に転送 |

GAM……………G-VRAMアクセスモード
　0：メインRAM
　1：G-VRAM

●拡張G-VRAMアクセスモードで，どこからのデータを移すかといったことを制御する．

| 番地 | I/O | 内　容 |
|---|---|---|
| 40 | O | ストローブポート |

MSB　LSB

| FBEEP | JOP1 | BEEP | GHSM | $\overline{\text{CLDS}}$ | CCK | CSTB | $\overline{\text{PSTB}}$ |
|---|---|---|---|---|---|---|---|

$\overline{\text{PSTB}}$……………プリンタストローブ
出力ポート10Hにセットされているプリンタへの出力データを出力するときに，このビットを1→0→1と変化させる

CSTB……………カレンダクロックストローブ
カレンダ時計にコマンドやデータを出力するとき，10Hポートにデータをセットし，このビットを0→1→0と変化させる

<table>
<tr><th>番地</th><th>I/O</th><th>内　容</th></tr>
<tr><td>40</td><td>O</td><td>CCK……………カレンダシフト用クロック<br>CLDS……………CRT インターフェース同期コントロール<br>GHSM……………グラフィックスハイスピードモード<br>0：高速モード<br>1：標準モード<br>BEEP……………ビープ音コントロール<br>0：BEEP OFF<br>1：BEEP ON<br>このビットを上げ下げすることによって，内蔵ブザー（カセットインターフェースのキャリアの音）を ON/OFF できる<br>JOP1……………汎用出力ポート<br>FBEEP……………サウンド出力</td></tr>
<tr><td>40</td><td>I</td><td>各種データ<br>MSB　　　　LSB
<table>
<tr><td></td><td></td><td>VRTC</td><td>CDI</td><td>SW2<br>S7</td><td>DCD</td><td>SHG</td><td>BUSY</td></tr>
</table>
BUSY……………プリンタ BUSY<br>0：プリンタ READY<br>1：プリンタ BUSY<br>現在プリンタが，データが受信できる状態かどうかを知ることができる<br>SHG……………高解像度 CRT モード<br>0：高解像<br>1：ノーマル<br>DCD……………RS-232C キャリアディテクト<br>SW2S7……………DISK ブートモード<br>CDI……………カレンダクロックからのデータ<br>VRTC……………垂直帰線期間信号</td></tr>
</table>

<table>
<tr><th>番地</th><th>I/O</th><th>内　容</th></tr>
<tr><td>52</td><td>O</td><td>
バックグランドカラー（デジタルモード時）<br>
MSB　　　　LSB
<table><tr><td>＼</td><td>BGG</td><td>BGR</td><td>BGB</td><td>＼</td><td>＼</td><td>＼</td><td>＼</td></tr></table>
●このポートを直接いじることにより，COLOR文などを使用しないでもバックグランドの色を変えることができる．<br>
　ただし，このポートの内容が有効なのは，512色表示モードではないときのみ．
</td></tr>
<tr><td>53</td><td>O</td><td>
画面の重ね合わせの制御<br>
MSB　　　　LSB
<table><tr><td>＼</td><td>＼</td><td>＼</td><td>＼</td><td>G2DS</td><td>G1DS</td><td>G0DS</td><td>TEXT DS</td></tr></table>
TEXT DS………TEXT 画面の表示<br>
0 ：ON<br>
1 ：OFF<br>
G0DS……………G-VRAM0（青）の表示<br>
0 ：ON<br>
1 ：OFF<br>
G1DS……………G-VRAM1（赤）の表示<br>
0 ：ON<br>
1 ：OFF<br>
G2DS……………G-VRAM2（緑）の表示<br>
0 ：ON<br>
1 ：OFF<br>
●このポートでは，グラフィック画面やテキスト画面をどのように重ねて表示するかが設定できる．SCREEN 文を使えばグラフィック画面の表示はON/OFFできるが，TEXT画面（キャラクタ画面）の表示（ON/OFF機能）は，このポートを直接いじることで可能
</td></tr>
</table>

| 番地 | I/O | 内　容 |
|---|---|---|
| 53 | O | になる．ただし，TEXT画面がON/OFFできるのはカラーグラフィックモード時のみ． |
| 54<br>∫<br>5B | O | カラーパレットの制御(デジタルモード時) |

MSB　　LSB

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | PG0 | PR0 | PB0 |

パレット番号0の色　54

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | PG1 | PR1 | PB1 |

パレット番号1の色　55

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | PG2 | PR2 | PB2 |

パレット番号2の色　56

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | PG3 | PR3 | PB3 |

パレット番号3の色　57

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | PG4 | PR4 | PB4 |

パレット番号4の色　58

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | PG5 | PR5 | PB5 |

パレット番号5の色　59

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | PG6 | PR6 | PB6 |

パレット番号6の色　5A

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | PG7 | PR7 | PB7 |

パレット番号7の色　5B

PGn……………パレット n 番の GREEN
PRn……………パレット n 番の RED
PBn……………パレット n 番の BLUE

| 番地 | I/O | 内　容 |
|---|---|---|
| 54<br>〜<br>5B | O | ●BASICのCOLOR文で指定するデジタルモード時でのパレットの割り付けを決めるポート．アナログ時も同じ54H〜5BHのポートを使用するが，値の指定のし方が異なる． |
| 54<br>〜<br>5B | O | **カラーパレットの制御（アナログモード時）** |

MSB　LSB

| 0 | 0 | PR02 | PR01 | PR00 | PB02 | PB01 | PB00 |
|---|---|---|---|---|---|---|---|

パレット番号0の赤，青の明るさのレベル（階調）を設定

| 0 | 1 | ＼ | ＼ | ＼ | PG02 | PG01 | PG00 |
|---|---|---|---|---|---|---|---|

パレット番号1の緑の明るさのレベルを設定
(55H〜5BHも同様)

PBn2〜PBn0…パレットn番の青の階調
PRn2〜PRn0…パレットn番の赤の階調
PGn2〜PGn0…パレットn番の緑の階調

●アナログモード時には，これらのポートは第6ビットが0のとき赤，青の階調，第6ビットが1のとき緑の階調を指定することになる．

| 番地 | I/O | 内　容 |
|---|---|---|
| 54 | O | **バックグランドカラーの設定（アナログモード時）** |

MSB　LSB

| 1 | 0 | BGR2 | BGR1 | BGR0 | BGB2 | BGB1 | BGB0 |
|---|---|---|---|---|---|---|---|

| 1 | 1 | ＼ | ＼ | ＼ | BGG2 | BGG1 | BGG0 |
|---|---|---|---|---|---|---|---|

BGB2〜BGB0…青の階調
BGR2〜BGR0…赤の階調
BGG2〜BGG0…緑の階調

●54Hポートは，第7ビットが1のときはアナログモードでのバックグランドカラーの指定になる．

| 番地 | I/O | 内　容 |
|---|---|---|
| 5C | O | **G-VRAMステイタス(独立アクセスモード時)**<br>MSB ～ LSB: [ / ] [ / ] [ / ] [ / ] [ / ] G2 G1 G0<br>G0……………G-VRAM0 (BLUE)<br>0：アクセス不可<br>1：アクセス可<br>G1……………G-VRAM1 (RED)<br>0：アクセス不可<br>1：アクセス可<br>G2……………G-VRAM2 (GREEN)<br>0：アクセス不可<br>1：アクセス可<br>●現在G-VRAMのどの画面がアクセス可能になっているかを知ることができる．独立アクセスモードのとき有効． |
| 5C<br>∫<br>5F | O | **アクセスするG-VRAMの選択(独立アクセスモード時)**<br>MSB ～ LSB<br>G-VRAM0 (BLUE) を選択　5C<br>MSB ～ LSB<br>G-VRAM1 (RED) を選択　5D<br>MSB ～ LSB<br>G-VRAM2 (GREEN) を選択　5E<br>MSB ～ LSB<br>メインRAMを選択　5F |

<table>
<tr><th>番地</th><th>I/O</th><th>内　容</th></tr>
<tr><td>5C<br>〜<br>5F</td><td>O</td><td>●これらのポートは，独立アクセスモード時にどのG-VRAMをアクセスするかを選択する．それぞれ出力するデータに関係なく，そのポートへ何らかのデータを出力することによって対応するG-VRAMが選択される．</td></tr>
<tr><td>70</td><td>I/O</td><td>オフセットアドレス<br><br>MSB　　　　　　　　　　　　　　　　LSB<br>
<table><tr><td>AP15</td><td>AP14</td><td>AP13</td><td>AP12</td><td>AP11</td><td>AP10</td><td>AP 9</td><td>AP 8</td></tr></table>
AP15〜AP 8……オフセットの上位バイト<br>●このポートは，テキストウィンドウ（8000H〜83FFH番地）に，メモリ空間のどの部分を映すかを指定する．またこの内容を読み出せば，現在テキストウィンドウに映されているのがメモリ空間のどこかを知ることもできる．</td></tr>
<tr><td>71</td><td>O</td><td>$N_{88}$-BASIC拡張ROM制御<br><br>MSB　　　　　　　　　　　　　　　　LSB<br>
<table><tr><td>$\overline{\text{EROM7}}$</td><td>$\overline{\text{EROM6}}$</td><td>$\overline{\text{EROM5}}$</td><td>$\overline{\text{EROM4}}$</td><td>$\overline{\text{EROM3}}$</td><td>$\overline{\text{EROM2}}$</td><td>$\overline{\text{EROM1}}$</td><td>$\overline{\text{IEROM}}$</td></tr></table>
$\overline{\text{EROM1}}$〜$\overline{\text{EROM7}}$………0：選択<br>1：選択しない<br>$\overline{\text{IEROM}}$…………………内部拡張ROM（E0ROM〜E3ROM）選択<br>0：選択<br>1：選択しない<br>●メモリ空間の6000H〜7FFFH番地の空間に割り当てられたバンクROMの選択をする．当然，一度に複数のビットを0にすることはできない．$\overline{\text{IEROM}}$が0の場合には拡張ROMとしてBASIC用拡張ROMのEROM 0〜EROM 3が選ばれる．また，このポートを読み出すことによって，現在どのバンクが選択されているかを知ることができる．</td></tr>
</table>

<table>
<tr><th>番地</th><th>I/O</th><th>内 容</th></tr>
<tr><td rowspan="2">78</td><td rowspan="2">O</td><td>オフセットアドレスインクリメント</td></tr>
<tr><td>MSB LSB<br><br>| ＼ | ＼ | ＼ | ＼ | SGS | B2 | B1 | B0 |<br><br>●出力する内容に関係なくこのポートへデータが出力されると，テキストウィンドウに映されるオフセットアドレス（70H番地参照）が1つ増やされる．</td></tr>
</table>

＊キーボードのとり込みに関しては本文第3章参照のこと．

# Z-80(Z-80A)
# ニーモニック↔機械語対照表

8ビット・ロード

| × | I | R | A | B | C | D | E | H | L | (HL) | (BC) | (DE) | (IX+d) | (IY+d) | (nn) | n |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LD A, × | ED 57 | ED 5F | 7F | 78 | 79 | 7A | 7B | 7C | 7D | 7E | 0A | 1A | DD 7E d | FD 7E d | 3A n n | 3E n |
| LD B, × | | | 47 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | | | DD 46 d | FD 46 d | | 06 n |
| LD C, × | | | 4F | 48 | 49 | 4A | 4B | 4C | 4D | 4E | | | DD 4E d | FD 4E d | | 0E n |
| LD D, × | | | 57 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | | | DD 56 d | FD 56 d | | 16 n |
| LD E, × | | | 5F | 58 | 59 | 5A | 5B | 5C | 5D | 5E | | | DD 5E d | FD 5E d | | 1E n |
| LD H, × | | | 67 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | | | DD 66 d | FD 66 d | | 26 n |
| LD L, × | | | 6F | 68 | 69 | 6A | 6B | 6C | 6D | 6E | | | DD 6E d | FD 6E d | | 2E n |
| LD (HL), × | | | 77 | 70 | 71 | 72 | 73 | 74 | 75 | | | | | | | 36 n |
| LD (BC), × | | | 02 | | | | | | | | | | | | | |
| LD (DE), × | | | 12 | | | | | | | | | | | | | |
| LD (IX+d), × | | | DD 77 d | DD 70 d | DD 71 d | DD 72 d | DD 73 d | DD 74 d | DD 75 d | | | | | | | DD 36 d n |
| LD (IY+d), × | | | FD 77 d | FD 70 d | FD 71 d | FD 72 d | FD 73 d | FD 74 d | FD 75 d | | | | | | | FD 36 d n |
| LD (nn), × | | | 32 n n | | | | | | | | | | | | | |
| LD I, × | | | ED 47 | | | | | | | | | | | | | |
| LD R, × | | | ED 4F | | | | | | | | | | | | | |

16ビット・ロード

| × | AF | BC | DE | HL | SP | IX | IY | nn | (nn) |
|---|---|---|---|---|---|---|---|---|---|
| LD AF, × | | | | | | | | | |
| LD BC, × | | | | | | | | 01 n n | ED 4B n n |
| LD DE, × | | | | | | | | 11 n n | ED 5B n n |
| LD HL, × | | | | | | | | 21 n n | 2A n n |
| LD SP, × | | | | F9 | | DD F9 | FD F9 | 31 n n | ED 7B n n |

ブロック転送

| | |
|---|---|
| LDI | ED A0 |
| LDIR | ED B0 |
| LDD | ED A8 |
| LDDR | ED B8 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| LD IX, × | | | | | | | | DD<br>21<br>n<br>n | DD<br>2A<br>n<br>n |
| LD IY, × | | | | | | | | FD<br>21<br>n<br>n | FD<br>2A<br>n<br>n |
| LD (nn), × | | ED<br>43<br>n<br>n | ED<br>53<br>n<br>n | 22<br>n<br>n | ED<br>73<br>n<br>n | DD<br>22<br>n<br>n | FD<br>22<br>n<br>n | | |
| PUSH × | F5 | C5 | D5 | E5 | | DD<br>E5 | FD<br>E5 | | |
| POP × | F1 | C1 | D1 | E1 | | DD<br>E1 | FD<br>E1 | | |

### ブロック・サーチ

| | |
|---|---|
| CPI | ED<br>A1 |
| CPIR | ED<br>B1 |
| CPD | ED<br>A9 |
| CPDR | ED<br>B9 |

### 8ビット算術論理演算

| × | A | B | C | D | E | H | L | (HL) | (IX +d) | (IY +d) | n |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ADD A, × | 87 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | DD<br>86<br>d | FD<br>86<br>d | C6<br>n |
| ADC A, × | 8F | 88 | 89 | 8A | 8B | 8C | 8D | 8E | DD<br>8E<br>d | FD<br>8E<br>d | CE<br>n |
| SUB × | 97 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | DD<br>96<br>d | FD<br>96<br>d | D6<br>n |
| SBC A, × | 9F | 98 | 99 | 9A | 9B | 9C | 9D | 9E | DD<br>9E<br>d | FD<br>9E<br>d | DE<br>n |
| AND × | A7 | A0 | A1 | A2 | A3 | A4 | A5 | A6 | DD<br>A6<br>d | FD<br>A6<br>d | E6<br>n |
| XOR × | AF | A8 | A9 | AA | AB | AC | AD | AE | DD<br>AE<br>d | FD<br>AE<br>d | EE<br>n |
| OR × | B7 | B0 | B1 | B2 | B3 | B4 | B5 | B6 | DD<br>B6<br>d | FD<br>B6<br>d | F6<br>n |
| CP × | BF | B8 | B9 | BA | BB | BC | BD | BE | DD<br>BE<br>d | FD<br>BE<br>d | FE<br>n |
| INC × | 3C | 04 | 0C | 14 | 1C | 24 | 2C | 34 | DD<br>34<br>d | FD<br>34<br>d | |
| DEC × | 3D | 05 | 0D | 15 | 1D | 25 | 2D | 35 | DD<br>35<br>d | FD<br>35<br>d | |

### CPUコントロール

| | |
|---|---|
| NOP | 00 |
| HALT | 76 |
| DI | F3 |
| EI | FB |
| IM 0 | ED<br>46 |
| IM 1 | ED<br>56 |
| IM 2 | ED<br>5E |

### 16ビット算術演算

| × | BC | DE | HL | SP | IX | IY |
|---|---|---|---|---|---|---|
| ADD HL, × | 09 | 19 | 29 | 39 | | |
| ADD IX, × | DD<br>09 | DD<br>19 | | DD<br>39 | DD<br>29 | |
| ADD IY, × | FD<br>09 | FD<br>19 | | FD<br>39 | | FD<br>29 |
| ADC HL, × | ED<br>4A | ED<br>5A | ED<br>6A | ED<br>7A | | |
| SBC HL, × | ED<br>42 | ED<br>52 | ED<br>62 | ED<br>72 | | |
| INC × | 03 | 13 | 23 | 33 | DD<br>23 | FD<br>23 |
| DEC × | 0B | 1B | 2B | 3B | DD<br>2B | FD<br>2B |

### エクスチェンジ

| | |
|---|---|
| EX AF, AF' | 08 |
| EX DE, HL | EB |
| EX (SP), HL | E3 |
| EX (SP), IX | DD<br>E3 |
| EX (SP), IY | FD<br>E3 |
| EXX | D9 |

### アキュムレータ操作

| | |
|---|---|
| DAA | 27 |
| CPL | 2F |
| NEG | ED<br>44 |
| CCF | 3F |
| SCF | 37 |

ローテート，シフト

| × | A | B | C | D | E | H | L | (HL) | (IX+d) | (IY+d) |
|---|---|---|---|---|---|---|---|---|---|---|
| RLC × | CB 07 | CB 00 | CB 01 | CB 02 | CB 03 | CB 04 | CB 05 | CB 06 | DD CB d 06 | FD CB d 06 |
| RRC × | CB 0F | CB 08 | CB 09 | CB 0A | CB 0B | CB 0C | CB 0D | CB 0E | DD CB d 0E | FD CB d 0E |
| RL × | CB 17 | CB 10 | CB 11 | CB 12 | CB 13 | CB 14 | CB 15 | CB 16 | DD CB d 16 | FD CB d 16 |
| RR × | CB 1F | CB 18 | CB 19 | CB 1A | CB 1B | CB 1C | CB 1D | CB 1E | DD CB d 1E | FD CB d 1E |
| SLA × | CB 27 | CB 20 | CB 21 | CB 22 | CB 23 | CB 24 | CB 25 | CB 26 | DD CB d 26 | FD CB d 26 |
| SRA × | CB 2F | CB 28 | CB 29 | CB 2A | CB 2B | CB 2C | CB 2D | CB 2E | DD CB d 2E | FD CB d 2E |
| SRL × | CB 3F | CB 38 | CB 39 | CB 3A | CB 3B | CB 3C | CB 3D | CB 3E | DD CB d 3E | FD CB d 3E |
| RLD | | | | | | | | ED 6F | | |
| RRD | | | | | | | | ED 67 | | |

| | A |
|---|---|
| RLCA | 07 |
| RRCA | 0F |
| RLA | 17 |
| RRA | 1F |

ジャンプ，コール，リターン

| × | UN COND | C | NC | Z | NZ | PE | PO | M | P | |
|---|---|---|---|---|---|---|---|---|---|---|
| JP ×, nn | C3 n n | DA n n | D2 n n | CA n n | C2 n n | EA n n | E2 n n | FA n n | F2 n n | |
| JR ×, e | 18 e−2 | 38 e−2 | 30 e−2 | 28 e−2 | 20 e−2 | | | | | |
| JP (HL) | E9 | | | | | | | | | |
| JP (IX) | DD E9 | | | | | | | | | |
| JP (IY) | FD E9 | | | | | | | | | |
| CALL ×, nn | CD n n | DC n n | D4 n n | CC n n | C4 n n | EC n n | E4 n n | FC n n | F4 n n | |
| DJNZ e | | | | | | | | | | 10 e−2 |
| RET × | C9 | D8 | D0 | C8 | C0 | E8 | E0 | F8 | F0 | |
| RETI | ED 4D | | | | | | | | | |
| RETN | ED 45 | | | | | | | | | |

リスタート

| | |
|---|---|
| RST 00H | C7 |
| RST 08H | CF |
| RST 10H | D7 |
| RST 18H | DF |
| RST 20H | E7 |
| RST 28H | EF |
| RST 30H | F7 |
| RST 38H | FF |

ビット操作

| × | A | B | C | D | E | H | L | (HL) | (IX +d) | (IY +d) |
|---|---|---|---|---|---|---|---|---|---|---|
| BIT 0, × | CB 47 | CB 40 | CB 41 | CB 42 | CB 43 | CB 44 | CB 45 | CB 46 | DD CB d 46 | FD CB d 46 |
| BIT 1, × | CB 4F | CB 48 | CB 49 | CB 4A | CB 4B | CB 4C | CB 4D | CB 4E | DD CB d 4E | FD CB d 4E |
| BIT 2, × | CB 57 | CB 50 | CB 51 | CB 52 | CB 53 | CB 54 | CB 55 | CB 56 | DD CB d 56 | FD CB d 56 |
| BIT 3, × | CB 5F | CB 58 | CB 59 | CB 5A | CB 5B | CB 5C | CB 5D | CB 5E | DD CB d 5E | FD CB d 5E |
| BIT 4, × | CB 67 | CB 60 | CB 61 | CB 62 | CB 63 | CB 64 | CB 65 | CB 66 | DD CB d 66 | FD CB d 66 |
| BIT 5, × | CB 6F | CB 68 | CB 69 | CB 6A | CB 6B | CB 6C | CB 6D | CB 6E | DD CB d 6E | FD CB d 6E |
| BIT 6, × | CB 77 | CB 70 | CB 71 | CB 72 | CB 73 | CB 74 | CB 75 | CB 76 | DD CB d 76 | FD CB d 76 |
| BIT 7, × | CB 7F | CB 78 | CB 79 | CB 7A | CB 7B | CB 7C | CB 7D | CB 7E | DD CB d 7E | FD CB d 7E |
| RES 0, × | CB 87 | CB 80 | CB 81 | CB 82 | CB 83 | CB 84 | CB 85 | CB 86 | DD CB d 86 | FD CB d 86 |
| RES 1, × | CB 8F | CB 88 | CB 89 | CB 8A | CB 8B | CB 8C | CB 8D | CB 8E | DD CB d 8E | FD CB d 8E |
| RES 2, × | CB 97 | CB 90 | CB 91 | CB 92 | CB 93 | CB 94 | CB 95 | CB 96 | DD CB d 96 | FD CB d 96 |
| RES 3, × | CB 9F | CB 98 | CB 99 | CB 9A | CB 9B | CB 9C | CB 9D | CB 9E | DD CB d 9E | FD CB d 9E |
| RES 4, × | CB A7 | CB A0 | CB A1 | CB A2 | CB A3 | CB A4 | CB A5 | CB A6 | DD CB d A6 | FD CB d A6 |
| RES 5, × | CB AF | CB A8 | CB A9 | CB AA | CB AB | CB AC | CB AD | CB AE | DD CB d AE | FD CB d AE |
| RES 6, × | CB B7 | CB B0 | CB B1 | CB B2 | CB B3 | CB B4 | CB B5 | CB B6 | DD CB d B6 | FD CB d B6 |
| RES 7, × | CB BF | CB B8 | CB B9 | CB BA | CB BB | CB BC | CB BD | CB BE | DD CB d BE | FD CB d BE |
| SET 0, × | CB C7 | CB C0 | CB C1 | CB C2 | CB C3 | CB C4 | CB C5 | CB C6 | DD CB d C6 | FD CB d C6 |
| SET 1, × | CB CF | CB C8 | CB C9 | CB CA | CB CB | CB CC | CB CD | CB CE | DD CB d CE | FD CB d CE |
| SET 2, × | CB D7 | CB D0 | CB D1 | CB D2 | CB D3 | CB D4 | CB D5 | CB D6 | DD CB d D6 | FD CB d D6 |
| SET 3, × | CB DF | CB D8 | CB D9 | CB DA | CB DB | CB DC | CB DD | CB DE | DD CB d DE | FD CB d DE |
| SET 4, × | CB E7 | CB E0 | CB E1 | CB E2 | CB E3 | CB E4 | CB E5 | CB E6 | DD CB d E6 | FD CB d E6 |
| SET 5, × | CB EF | CB E8 | CB E9 | CB EA | CB EB | CB EC | CB ED | CB EE | DD CB d EE | FD CB d EE |
| SET 6, × | CB F7 | CB F0 | CB F1 | CB F2 | CB F3 | CB F4 | CB F5 | CB F6 | DD CB d F6 | FD CB d F6 |
| SET 7, × | CB FF | CB F8 | CB F9 | CB FA | CB FB | CB FC | CB FD | CB FE | DD CB d FE | FD CB d FE |

入　力

| | |
|---|---|
| IN A, n | DB n |
| IN A, (C) | ED 78 |
| IN B, (C) | ED 40 |
| IN C, (C) | ED 48 |
| IN D, (C) | ED 50 |
| IN E, (C) | ED 58 |
| IN H, (C) | ED 60 |
| IN L, (C) | ED 68 |
| INI | ED A2 |
| INIR | ED B2 |
| IND | ED AA |
| INDR | ED BA |

出　力

| | |
|---|---|
| OUT n, A | D3 n |
| OUT (C), A | ED 79 |
| OUT (C), B | ED 41 |
| OUT (C), C | ED 49 |
| OUT (C), D | ED 51 |
| OUT (C), E | ED 59 |
| OUT (C), H | ED 61 |
| OUT (C), L | ED 69 |
| OUTI | ED A3 |
| OTIR | ED B3 |
| OUTD | ED AB |
| OTDR | ED BB |

引用資料：NEC技術資料μCOM-82インストラクション活用表（IEM-656A　JAN-12-80）

# 10進数↔16進数変換表

| 下位 上位 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 1 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 2 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 3 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| 4 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| 5 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| 6 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| 7 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |
| 8 | 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 |
| 9 | 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 |
| A | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 |
| B | 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 |
| C | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 |
| D | 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 | 220 | 221 | 222 | 223 |
| E | 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 |
| F | 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 | 248 | 249 | 250 | 251 | 252 | 253 | 254 | 255 |

# 1バイト符号付16進数

| 上位＼下位 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 1 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 2 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 3 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| 4 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| 5 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| 6 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| 7 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |
| 8 | −128 | −127 | −126 | −125 | −124 | −123 | −122 | −121 | −120 | −119 | −118 | −117 | −116 | −115 | −114 | −113 |
| 9 | −112 | −111 | −110 | −109 | −108 | −107 | −106 | −105 | −104 | −103 | −102 | −101 | −100 | −99 | −98 | −97 |
| A | −96 | −95 | −94 | −93 | −92 | −91 | −90 | −89 | −88 | −87 | −86 | −85 | −84 | −83 | −82 | −81 |
| B | −80 | −79 | −78 | −77 | −76 | −75 | −74 | −73 | −72 | −71 | −70 | −69 | −68 | −67 | −66 | −65 |
| C | −64 | −63 | −62 | −61 | −60 | −59 | −58 | −57 | −56 | −55 | −54 | −53 | −52 | −51 | −50 | −49 |
| D | −48 | −47 | −46 | −45 | −44 | −43 | −42 | −41 | −40 | −39 | −38 | −37 | −36 | −35 | −34 | −33 |
| E | −32 | −31 | −30 | −29 | −28 | −27 | −26 | −25 | −24 | −23 | −22 | −21 | −20 | −19 | −18 | −17 |
| F | −16 | −15 | −14 | −13 | −12 | −11 | −10 | −9 | −8 | −7 | −6 | −5 | −4 | −3 | −2 | −1 |

## 参考文献


●「PC-8801mkIISRユーザーズマニュアル」 NEC
●「PC-Technow8800mkII」平松，八木共著　システムソフト社
●「PC-8801mkIISRテクニカルマップI ソフトウェアを知る」小林，今野，平間共著　秀和システムトレーディング(株)
●「電子科学」 1983年　1月号　産報出版
●「Oh PC」1983年　11月号，1984年　3月号　日本ソフトバンク社

# 索引

## 数字



## アルファベット

## あ行



## か行



## さ行

## た行



## な行



## は行



## ま行



## ら行



## わ行

前田　光男（まえだ　みつお）
昭和27年6月10日生れ．東京理科大学理学部卒．現在，東京都江東区立砂町中学校教諭．同校で科学部の顧問をし，生徒にパーソナルコンピュータを指導中．著書に「マシン語でゲームを作る本」「mkII FAN BOOKS[2]マシン語入門」（技術評論社）．

PC-8801mkIISR
SR SUPER BOOKS[2]
やさしいマシン語

昭和60年8月20日　初版　第1刷発行

著　者　前田　光男
発行者　片岡　巌
発行所　株式会社　技術評論社
東京都千代田区平河町1-4-12
電話　03(262)9351　営業部
03(237)8315　編集部
印　刷
製　本　加藤文明社

定価はカバーに表示してあります．



ISBN4-87408-279-3 C3055

SUPER BOOKS

第1章
初級者から中級者までのマシン語

第2章
モノクロ画面からカラー画面への招待

第3章
キーボードを操作する

第4章
グラフィックスの世界

第5章
スムーズ・スクロール

マシン語アラカルト

ISBN4-87408-279-3 C3055 ¥2000E 定価2,000円